# DESIGN AND IMPLEMENTATION OF AN OPEN-SOURCE OPTICALLY STIMULATED LUMINESCENCE MEASUREMENT SYSTEM

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

DİREN MARABA

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
PHYSICS

JULY 2017

Approval of the thesis:

**DESIGN AND IMPLEMENTATION OF AN OPEN-SOURCE OPTICALLY STIMULATED LUMINESCENCE MEASUREMENT SYSTEM**

submitted by **DİREN MARABA** in partial fulfillment of the requirements for the degree of **Master of Science in Physics Department, Middle East Technical University** by,

Prof. Dr. Gülbin Dural Ünver
Dean, Graduate School of **Natural and Applied Sciences** _____

Prof. Dr. Altuğ Özpineci
Head of Department, **Physics** _____

Prof. Dr. Enver Bulur
Supervisor, **Physics Dept., METU** _____

**Examining Committee Members:**

Prof. Dr. Ahmet Oral
Physics Dept., METU _____

Prof. Dr. Enver Bulur
Physics Dept., METU _____

Prof. Dr. Nizami Hasanli
Physics Dept., METU _____

Assoc. Prof. Dr. Mustafa Hicabi Bölükdemir
Physics Dept., Gazi University _____

Assist. Prof. Dr. Emre Yüce
Physics Dept., METU _____

**Date:** 07/07/2017

**I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.**

Name, Last name: DİREN MARABA

Signature:

## ABSTRACT

### DESIGN AND IMPLEMENTATION OF AN OPEN-SOURCE OPTICALLY STIMULATED LUMINESCENCE MEASUREMENT SYSTEM

Maraba, Diren

M.Sc., Department of Physics

Supervisor: Prof. Dr. Enver Bulur

July 2017, 145 Pages

Optically Stimulated Luminescence (OSL) is the light emission from an irradiated solid (insulator or a wide band gap semiconductor) upon illumination with light of suitable wavelength. Although the phenomenon has been known for a long time, OSL has emerged as a practically applicable dosimetry technique in the past two decades. Recently introduced materials like alumina and beryllia have found use in the field of radiation dosimetry. The purpose of this study is to design and construct a simple multi-sample OSL measurement system. The device is realized using open-source hardware and software platforms which reduced the development costs to very low levels. The system is able to measure OSL from materials relevant for dosimetry and dating. The measurement system is based on the Arduino DUE microcontroller board. The control of the sample positioning mechanism, stimulation system, precise timing and photon counting is achieved using this board. The control software on the client computer is written in Python language which can be run on any platform. The stimulation system is based on high power light emitting diode (LED) in visible region of the electromagnetic spectrum and three commonly used stimulation modes, continuous wave (CW-), linearly modulated (LM-) and

time-resolved (TR-) OSL modes are achievable. Luminescence emission is measured using a bialkali photomutiplier tube (PMT) module which works in photon counting mode and produces pulses as output. The pulses from the PMT module is counted using a fast timer counter of the microcontroller. The optical characteristics of system is measured and presented. In addition, the OSL measurement system is tested using materials such as $Al_2O_3$:C, BeO, Quartz, $ZrSiO_4$, which are used for dosimetry and dating studies.

**Keywords:** Optically Stimulated Luminescence (OSL), Luminescence Dosimetry, Scientific Instrumentation.

# ÖZ

## OPTİK UYARMALI LUMİNESANS ÖLÇÜM SİSTEMİNİN AÇIK KAYNAKLI TASARIMI VE GERÇEKLEŞTİRİLMESİ

Maraba, Diren
Yüksek Lisans, Fizik Bölümü
Tez Yöneticisi: Prof. Dr. Enver Bulur

Temmuz 2017, 145 Sayfa

Optik Uyarmalı Lüminesans (OSL) ışınlanmış katı maddenin (yalıtkan ya da geniş bant aralıklı yarı iletken) uygun dalga boyu ile uyarılması sonucu ışık yayılımıdır. Bu olgu uzun süreden beri bilinmesine rağmen, OSL son yirmi yıl içerisinde pratik olarak uygulanabilen bir dozimetri tekniği haline gelmiştir. Alumina ve beryllia gibi yakın zamanda önerilen malzemeler radyasyon dozimetrisi alanında kullanım alanı bulmuştur. Bu çalışmanın amacı basit bir çok örnekli OSL ölçüm cihazı tasarlamak ve imal etmektir. Açık kaynak kodlu yazılım ve donanım kullanarak geliştirme maliyetlerini düşürmesi cihazın en belirgin özelliğidir. Sistem dozimetri ve yaş tayini çalışmalarında uygun materyalleri ölçebilme yeteneğine sahiptir. Ölçüm sisteminin merkezinde yer alan Arduino DUE mikro denetleyici kartı; numune değiştirme mekanizması ve uyarım sistemi kontrolü ile hassas zamanlama ve foton sayma işlemlerini yapmaktadır. İstemci bilgisayardaki kullanıcı ara yüzü her platformda kullanılması amaçlanarak Python dilinde yazılmıştır. Elektromanyetik tayfın görünür bölgesinde çalışan yüksek güçlü ışık yayan diyotları temel alan uyarım sistemi, sürekli dalga (CW-), doğrusal kiplenimli (LM-) ve zaman çözünürlüklü (TR-) OSL kiplerini desteklemektedir. Lüminesans yayımı, foton

sayma kipinde çalışan bir bialkali katotlu fotoçoğaltıcı (PMT) modül ile ölçülmüştür. Modülden gelen pulslar, Arduino kartında gerçekleştirilen hızlı sayıcı ile kaydedilmiştir. Sistemin optik ve elektronik karakteristik ölçümleri gerçekleştirilerek; OSL ölçüm sistemi dozimetri ve yaş tayini çalışmalarında kullanılan $Al_2O_3$:C, BeO, Quartz, $ZrSiO_4$ gibi malzemelerle de test edilmiştir.

**Anahtar Kelimeler:** Optik Uyarmalı Lüminesans (OSL), Lüminesans Dozimetri, Bilimsel Enstürmantasyon.

*To scientists throughout history having paid the price for their studies*

## ACKNOWLEDGEMENTS

First, I would like to express my deep gratitude to my supervisor Prof. Dr. Enver Bulur for his guidance, encouragement and especially his continuous patience not only through this study, but also through all my graduate education period at METU.

I would like to thank my mother Hülya Alkan and my father Fikret Maraba for their love, patience, and support. I would also like to thank Gül Deniz Hoş for being in my life with her limitless support and understanding during my graduate study. I could always overcome the most troublesome and tiring time to the help of them. Thus, without them, none of what I have achieved would have been possible.

I wish to thank technician Hakan Sağ from Department of Physics for his help during mechanical construction of the measurement system and my friend Uğur Şahin for his help and comments on software programming.

Last but not the least, I would like to thank all my friends for their endless support.

## TABLE OF CONTENTS

ABSTRACT ..... v
ÖZ ..... vii
ACKNOWLEDGEMENTS ..... x
TABLE OF CONTENTS ..... xi
LIST OF FIGURES ..... xiii
LIST OF TABLES ..... xvii
CHAPTERS
1. INTRODUCTION ..... 1
2. LUMINESCENCE ..... 7
2.1 Luminescence Production ..... 7
2.2 Optically Stimulated Luminescence (OSL) ..... 10
2.2.1 Model and Equations ..... 11
2.2.2 Stimulation Modes of OSL ..... 17
2.2.2.1 Continuous-Wave Optically Stimulated Luminescence ..... 19
2.2.2.2 Linearly-Modulated Optically Stimulated Luminescence ..... 21
2.5.1.1 Pulsed Optically Stimulated Luminescence ..... 22
2.3 OSL Dosimetry ..... 25
3. SCIENTIFIC INSTRUMENTATION ..... 29
3.1 Instrumentation Basics ..... 29
3.1.1 Characteristics of an Instrumentation System ..... 32
3.2 Optically Stimulated Luminescence Measurement Technology ..... 33
3.2.1 Light Detection Unit ..... 35
3.2.1.1 Photomultiplier Tubes ..... 35
3.2.1.2 Other Detectors ..... 37
3.2.2 Stimulation Sources ..... 37
3.3 Open-Source Hardware and Software ..... 39
3.3.1 Microcontrollers ..... 39
3.3.2 Arduino Development Platform ..... 40

3.3.2.1 Software and Communication of Arduino ..... 41
3.2.3 Python Programming Language ..... 42
4. DESIGN AND CONSTRUCTION OF THE AUTOMATED OSL READER ..... 45
4.1 Structure and Properties of the OSL Reader..... 45
4.2 Components and Parts of the OSL Reader .....47
4.2.1 Measurement Chamber ..... 47
4.2.2 Motorized Sample Changer Unit..... 49
4.2.3 Electronic Control Unit ..... 50
4.2.3.1 Hardware and Connections ..... 51
4.2.3.2 Software ..... 52
4.2.4 User Interface (PC Software) ..... 58
4.3 Principles Measurement Modes ..... 62
4.3.1 Continuous-Wave Optically Stimulate Luminescence..... 62
4.3.2 Linearly-Modulated Optically Stimulate Luminescence..... 62
4.3.3 Time-Resolved Optically Stimulate Luminescence..... 62
5. RESULTS AND DISCUSSION..... 63
5.1 Determination of the Stimulation LED Characteristics ..... 63
5.2 Determination of Measurement Chamber Characteristics..... 68
5.3 Test Experiments..... 72
5.3.1 CW-OSL Mode..... 75
5.3.2 LM-OSL Mode ..... 80
5.3.3 TR-OSL Mode ..... 85
6. SUMMARY AND CONCLUSIONS..... 87
REFERENCES ..... 91
APPENDICES
A. WIRING DIAGRAMS ..... 101
B. GENERAL VIEW OF THE OSL MEASUREMENT SYSTEM ..... 109
C. MICROCONTOLLER SOFTWARE CODE ..... 115
D. USER INTERFACE SOFTWARE CODE..... 127

## LIST OF FIGURES

FIGURES

Figure 2.1: A simplified sketch of luminescence mechanism ..... 7

Figure 2.2: Energy band model description of OSL phenomenon ..... 11

Figure 2.3 Schematics of an OSL measurement system with the fundamental elements ..... 18

Figure 2.4: A summary of three main OSL stimulation modes, namely CW-OSL, LM-OSL and POSL ..... 19

Figure 2.5: An example of a CW-OSL decay (irradiated BeO) ..... 20

Figure 2.6: An example of a LM-OSL (Al2O3:C) ..... 22

Figure 2.7: A example of TR-OSL (BeO) ..... 25

Figure 2.8: An example of dose response trend line of BeO..... 26

Figure 3.1: Measurement and Control System ..... 30

Figure 3.2: A diagram of measurement system elements ..... 31

Figure 3.3: A simplified diagram of setup for measuring OSL..... 34

Figure 3.4: The schematic diagram of a PMT. $U_c$ is cathode voltage, $U_d$ denotes the potential difference between two dynodes, R is resistor..... 35

Figure 4.1: Simplified block diagram of the automated OSL reader ..... 46

Figure 4.2: Simplified sketch of measurement chamber ..... 48

Figure 4.3: Simplified sketch motorized sample changer unit ..... 50

Figure 4.4: Block diagram of hardware of electronic control unit ..... 51

Figure 4.5: Flow chart of motorized sample changer unit part of the software ..... 53

Figure 4.6: Flow chart of OSL measurement part of the software..... 56

Figure 4.7: Screenshot of user interface software ..... 59

Figure 4.8: Flow chart of User Interface Software ..... 61

Figure 5.1: Emission spectra of the stimulation LED at different current values ... 64

Figure 5.2: Output power of LED with respect to current passing through ..... 65

Figure 5.3: The PCB temperature of the LED after 10 minutes of operation time with respect to current passing through. Inset: The relationship between voltage applied to the junction and current passing through ..... 66

Figure 5.4: Output Signal of Photodiode in Time for a LED Pulse ..... 67

Figure 5.5: Output power of the LED on sample with respect to duty cycle ..... 68

Figure 5.6: A simplified sketch of the experimental setup used for reflection and transmission measurements ..... 69

Figure 5.7: Transmission spectrum of dichroic mirror (solid-line) and emission spectrum of LED (dashed-line). [Transmission spectrum of the dichroic mirror is measured at 45° incident angle] ..... 70

Figure 5.8: Reflection spectrum of dichroic mirror (black/square). Transmission spectrum of the UV filter pack (purple/triangle). Transmission spectrum of dichroic mirror and UV filter pack combination (red/circle). [Reflection and transmission spectra of the dichroic mirror are measured at 45° incident angle] ..... 71

Figure 5.9: Background measurement of OSL measurement system when stimulation light is on and off ..... 72

Figure 5.10: A picture of samples used for OSL measurements ..... 73

Figure 5.11: Decay curves of $Al_2O_3$:C (100 mGy irradiated) taken from consecutive measurements [(line) is first measurement, (dashed) is second measurement]. Inset: Dose response of $Al_2O_3$:C in the range from 0.1 Gy to 1 Gy. ..... 75

Figure 5.12: Decay curves of BeO chips with different doses: 0.50 Gy (black/square), 0.25 Gy (red/circle), 0.10 Gy (blue/triangle), 0.05 Gy (magenta/reverse triangle) and background (green/diamond). Inset: Dose response of BeO chips in the interval 5 mGy to 500 mGy ..... 76

Figure 5.13: Decay curves of quartz (100 Gy irradiated) taken from consecutive measurements [(line) is first measurement, (dash) is second measurement]. Inset: Dose response of quartz from 50 Gy to 250 Gy ..... 78

Figure 5.14: CW-OSL signal from zircon samples fitted with combination of three exponential decay curves. Inset: Decay components of the luminescence signal: Fast Decay (dashed/blue line), Medium Decay (solid/black line) and Slow Decay (dotted/red line) ..... 79

Figure 5.15: Decay curves of zircon samples with different doses: 5 Gy (black/square), 10 Gy (blue/triangle), 25 Gy (red/circle) and background (green/dashed). Inset: Dose response of zircon samples chips in the interval 1 Gy to 50 Gy ..... 80

Figure 5.16: LM-OSL data of 0.5 Gy irradiated $Al_2O_3$:C chip with fitted curve and its components (dotted, dashed and dot-dashed lines). Inset: LM-OSL curves of $Al_2O_3$:C chips with various doses together with background ..... 81

Figure 5.17: LM-OSL data of 0.5 Gy irradiated BeO chip with fitted curve and its components (dotted, dashed, dot-dashed and green-solid). Inset: LM-OSL curves of BeO chips irradiated at 250 mGy and 500 mGy together with measurement background ..... 82

Figure 5.18: LM-OSL data of 100 Gy irradiated quartz grains with fitted curve and its components (dotted/blue, dashed/green, solid/magenta and dot-dashed/cyan lines). Inset: LM-OSL curves of quartz grains irradiated with 50 Gy, 75 Gy and 100 Gy ..... 83

Figure 5.19: LM-OSL data of 10 Gy irradiated zircon sample with fitted curve and its components (solid/blue, dotted/green and dashed/magenta lines). Inset: LM-OSL curves of zircon samples irradiated with 5 Gy, 10 Gy, 25 Gy and background measurement ..... 85

Figure 5.20: TR-OSL data of 500 mGy irradiated $Al_2O_3$:C chip with fitted curve. Inset: Dose response of TR-OSL measurements of $Al_2O_3$:C ..... 86

Figure A.1: Pinout Diagram of Arduino DUE microcontroller board ..... 102

Figure A.2: Wiring and connection diagram of power supplies ..... 104

Figure A.3: Wiring and connection diagram of front panel indicators and switches ..... 105

Figure A.4: Wiring and connection diagram of sample tray and sensor ..... 106

Figure A.5: Wiring and connection diagram of stimulation LED and PMT module ..... 107

Figure B.1: General view of the OSL measurement system ..... 110

Figure B.2: View of measurement chamber ..... 111

Figure B.3: Top view of the OSL measurement system ..... 112

Figure B.4: Front view of OSL measurement system ..... 113

Figure B.5: Back view of OSL measurement system ..... 114

## LIST OF TABLES

TABLES

Table 2.1: Luminescence types and excitation sources ..... 9

Table 4.1: List of the commands implemented in the software for the sample changer unit..... 54

Table 4.2: List of the commands implemented in the software for the sample changer unit ..... 57

Table 5.1: Annealing temperature and durations of samples before measurements ..... 73

Table 5.2: Functions used for curve-fitting in different OSL modalities ..... 74

Table A.1: Input and Outputs Pins of the Microcontroller ..... 103

# CHAPTER 1

## INTRODUCTION

Luminescence is a term, which is first introduced by Eilhard Wiedemann (1888), refers to spontenous emission of the photons from a material, where this emission is not a resultant of heating. When a charged particle or combination of particles (such as atoms, molecules) are excited to a higher energy state, they are inclined to return initial lower energy state by releasing energy in order to reach the stability of equilibrium condition. There are two possible ways of these transition called as radiative and non-radiative transitions. If the transition is radiative, emission of photons is observed. This process fundamentally describes the luminescence phenomena. Luminescence may also be observed as a result of stimulating an insulator or semiconductor material which was previously excited by an ionizing radiation. There are different types of luminescence processes. Most well knowns can be listed as follows: Chemiluminescence (emission due to chemical reaction process), Electroluminescence (emission due to electric current), Radioluminescence (emission due to irradiation of material) Thermally Stimulated Luminescence or Thermoluminescence (emission due to recombination of electron traps as a result of heating) and Optically Stimulated Luminescence (emission due to recombination of trapped electrons as a result of stimulation with light).

Optically Stimulated Luminescence (OSL) is a prevalently used technique in personal, environmental and clinical dosimetry since the technique enables fast readout with high efficiency and it is reproducible. The term is used for the luminescence emission from an irradiated material (insulator or semiconductor) due to light exposure. Even though, Antonov-Romanovskii et al. (1956) first offered

radiation dose measurement using OSL, usage of the technique reported almost 12 years later by Bräunlich et al. (1967) and, Sanborn and Beard (1967). It had been realized that phosphors they used (MgS, CaS, SrSe doped with Ce, Sm, Eu and various other rare earth elements) are not suitable for luminescence dosimetry. Later, OSL measurements on naturally occurring materials such as quartz and feldspar samples (Huntley et al., 1985 using an argon-ion laser for stimulation) have initiated the studies on age determination of archeological and geological findings. The development of technology (and production facilities) for stimulation sources (i.e. filtered lamps, light emitting diodes and solid state lasers) have enabled OSL to become a significant technique for radiation dosimetry, dating and material research. Introduction of $Al_2O_3$:C -a material with very good luminescence characteristics (see Markey et al., 1995)- has boosted the applications in the field of radiation dosimetry. Besides, research on red laser stimulable materials such as BaFBr:Eu have enabled the usage of the technique (known also as photo stimulated luminescence, PSL) for X-ray detection (Lakshmanan, 1996). Two dimensional detectors (image plates) based on BaFBr:Eu has enabled the direct measurements of X-ray images known as "computed radiography" (von Seggern, 1999). Extensive reviews for OSL technique including fundamental concepts and many applications have been presented by Bøtter-Jensen et al. (2003) and Yukihara and McKeever (2011).

$Al_2O_3$:C (see e.g. McKeever et al., 1996; Yukihara, 2014) and BeO (see e.g. Bulur and Göksu, 1998; Sommer and Henniger, 2006, Jahn et al., 2013) are only two commercialized OSL dosimeters as of today. Nowadays, the usage and applications of these materials have been greatly extended. Radiation exposure of millions, who are called radiation workers, are monitored by these passive detectors utilizing OSL technique. The use of dosimeters is investigated in hospitals as a part of quality control programs (Lovelock et al., 2012). There are some clinical trials that radiotherapy is monitored using OSL (Aguirre et al., 2009). The radiation exposure of astronauts was also monitored using OSL by both U. S. National Aeronautics and Space Administration (NASA) (Zhou et al., 2009) and European Space Agency (ESA) (Berger et al., 2016). More information regarding the usage of

$Al_2O_3$:C and BeO as dosimeters can be found in Yukihara et. al (2014) and Yukihara et. al (2016) respectively. There is a great increase in the usage of these materials, there is still a need for research on the existing and new materials.

Measurement of many samples in a minimum possible time is a demand for routine dosimetry and dating studies. For this reason, automated OSL readers have been introduced and they are extensively in use around the globe (see e.g. Yukihara and McKeever, 2011). It is essential for automated OSL readers to have a compact and stable measurements system. Moreover, they generally include a sample changer and luminescence measurement units which are controlled and monitored by a personal computer. First automated OSL reader, Elsec 9010 OSL system, is developed by Littlemore Scientific Engineering, UK at the beginning of 1990s. The development of automated OSL reader are depicted by Bøtter-Jensen (1997), Bøtter-Jensen et al. (2000) and Bortolot (2000). The Risø TL/OSL reader (Risø National Laboratory, Denmark, http://www.usu.edu/geo/luminlab/Reader.pdf, Accessed on May19$^{th}$, 2017), Daybreak TL/OSL readers (Daybreak Nuclear and Medical Systems, Inc., CT, USA, http://www.daybreaknuclear.us/daybreak_frameset.html, Accessed on May19$^{th}$, 2017), and the Freiberg Instruments lexsyg TL/OSL reader (Freiberg Instruments, Germany, http://www.lexsyg.com/tlosl-reader.html, Accessed on May 19$^{th}$, 2017) are other examples of commercial instruments which are still in use today. These commercial readers are widely preferred for scientific research due to their precision of measurements and compact instrumentation. Lately revealed models enable users to perform a sequence of irradiation and measurement process on different samples with different modes successively. Even though there are a lot of advantages of commercial readers which cannot go unnoticed, their abilities (which are open for users) are limited to the ones that are defined by manufacturer and cannot be easily modified by users. Especially for scientific research, it might be crucial to modify an existing equipment to perform novel experiments. Thus, commercial instruments may sometimes be disadvantageous for researchers. In addition, the cost of these compact instruments makes a purchase prohibitive for some research laboratories. For these reasons, there has been efforts on producing cost-efficient home-made luminescence measurement

devices (or readers). One of them is lately developed in South Korea by Choi et al. (2014). They designed a compact and economical OSL reader which may be load up to 12 samples and includes an X-ray generator for irradiating samples and two blues light emitting diodes (LED) for optical stimulation. Another one is presented by Guérin and Lefèvre (2014). They designed an automated instrument which is capable of performing thermoluminescence and optically stimulated luminescence measurements. Kearfott and West (2015) also represented a flexible low-cost optically stimulated luminescence reader utilizing multiple excitation wavelengths. Previously, a low-cost OSL measurement system using a green LED for stimulation was also reported (Maraba and Bulur, 2017). In this manuscript, an enhanced version of that measurement system is presented.

Another significant point for the research laboratories focused on the basics of the luminescence mechanisms is being able to control the instrumentation in both hardware and software aspects. The development of new measurement protocols and measurement techniques are possible by having this ability. Commercial luminescence measurement instruments are generally not offering such flexibility. It is because not these instruments are unadaptable but they are basically commercialized. On the other hand, use of non-commercial, homemade developed automated OSL reader may sometimes be more beneficial for research laboratories for multi-sample measurements. Since, access to hardware and software enables the device be open for modifications (i.e. attaching different measurement accessories such as monochromators, filter wheels, etc.) when necessary.

There are various types of developer packages available for writing the control software for an instrument, including *LabVIEW*, *Daisy Lab*, etc. For instance, National Instruments *LabVIEW* is very popular and beneficial software for scientific instrumentation. Users can create unique measurement control software over several instruments and combinations of them. However, users still have to pay a considerably high software license. On the other hand, the open-source movement is increasingly providing unlimited access to software and even hardware in almost every area for the last two decades. Advantage of the open-source software and hardware is that the license that covers product or code allow users to study, modify

and even redistribute the new version of it. Most of the software are available at zero cost and the hardware are cheaper than commercialized ones since anyone can manufacture as long as they have ability to do so (Harland and Forster, 2012). Hence, the movement aroused interest of people including researchers. It is non-negligible that the possibility of building scientific instruments using open-source software and hardware reduces the cost of research (see Pearce, 2014). Among the various alternatives, *Python* (Rossum, 1995) is an open source language that is used frequently in high-level software programming. It has become popular among those who is dealing with instrumentation. Hughes (2011) described essentials of instrumentation using *Python* in his book. There are also some examples of usage of *Python* in different areas of physics (see Borcherds, 2007, Greenfield, 2011, Imreh, 2014, Koenka et al., 2014).

Moreover, most of the equipment in use today are based on microcontrollers since they are good at recording simple parameters and controlling events. There are also publicly available hardware designs (known as free and open-source hardware) in significant amount (Pearce, 2014). Arduino, which is a microcontroller with an open-hardware and software platform (supported by a large community), has become a very significant replacement for expensive scientific instrumentation and research equipment (Fisher and Gould, 2012).

The main purpose of this study is to design and construct a low-cost, open hardware and software based automated OSL reader. The reader presented in this thesis is neither comparable with the ones commercially available nor constructed for this purpose. This thesis aims to be an example of open source hardware and software utilization for an automated OSL reader construction (which has not been done before). In this manuscript, design and implementation considerations are given together with the test experiments carried out using materials relevant for dosimetry and dating.

In Chapter 2, the theoretical aspects of OSL mechanism are discussed. There are three different stimulation modes available for users in the instrument: Continuous Wave OSL (CW-OSL), Linearly Modulated OSL (LM-OSL) and Time-

Resolved OSL (TR-OSL). These modes are portrayed briefly in addition to description of simple model to explain the luminescence production mechanisms.

In Chapter 3, some background information about scientific instrumentation is given. OSL measurement technology in terms of usage of detectors and stimulation sources discussed in detail. Finally, open software and hardware movement is discussed specific to Arduino and *Python* environment.

In Chapter 4, the design and development of an automated OSL reader, which consist of computer software, electronic hardware and firmware parts, is explained. Detailed software flowcharts are given in order to visualize the working logic of measurement system.

In Chapter 5, optical properties and structure of the measurement setup is mentioned in detail. The results of characteristics measurements for stimulation light and measurement chamber are given. In addition, OSL measurement results of test experiments performed on $Al_2O_3$:C, BeO, quartz ($SiO_2$) and natural zircon ($ZrSiO_4$) samples are given together with a brief discussion of OSL signals.

In Chapter 6, an overall conclusion of the research is presented.

# CHAPTER 2

# LUMINESCENCE

## 2.1 Luminescence Production

Radiative emission processes in solids resulting from transitions of electrons from a high energy level to a lower energy level is called luminescence. It is called as cold light or glow phenomena and must not be confused with the black body emission of a hot object. As it can be seen from Figure 2.1, a generalized picture of a luminescence mechanism can be drawn using a two level system. Electrons in the ground state can be excited to the higher energy state as a result of energy absorption followed by spontaneous relaxation to the ground state with a photon emission.

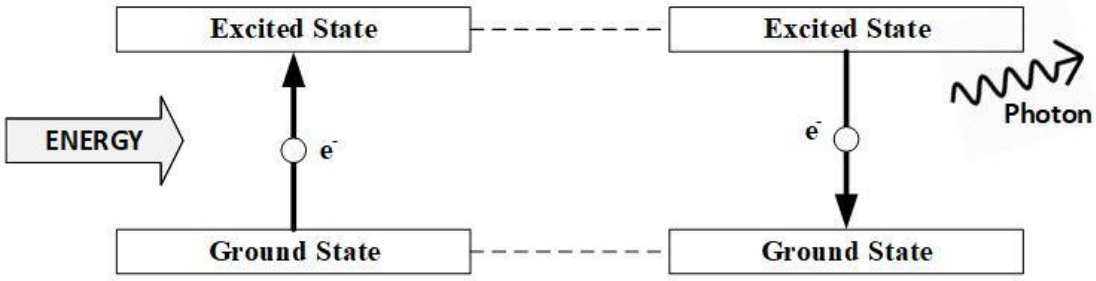



**Figure 2.1:** A simplified sketch of luminescence mechanism. On the left side, electron movement to excited state as a result of energy absorption is drawn. On the right side, the emission of luminescence as a result of electron relaxation is shown.

In theory, energy of absorbed and emitted photon from an atom or an ion is equal to the energy difference between states that transition of electrons occurs. Nevertheless, it is almost impossible to find a single isolated particle in nature. For the case of crystalline solids, interaction of molecules and atoms are inevitable. Electronic band structure is as a result of these interactions. According to the band theory of solids, there is region called as an energy gap between conduction band and valence band. This region is known as forbidden region because there are no energy states exists in a perfect crystalline structure. However, various types of defects in the crystal is observed very frequently. These defects can be listed as point, line and volume defects in three major types. Point defects are observed in the atomic level as a result of absence of an atom or substitution in a lattice. On the other hand, line and volume defect are called to be alterations in the molecular level. Various types of point defect are observed in solid. Some of the point defects observed in the solids can be listed as Schottky defects (absence of cation and anion, vacancy in non-ionic crystal), anion and cation vacancies, Frenkel defects (cation vacancy plus same cation as interstitial) and impurity atoms (see Chapter 30 of Ashcroft and Mermin, 1976 for more detailed information). Occurrence of different energy states in the forbidden gap become possible due to these defects. Therefore, an electron or a hole (the lack of an electron) may be captured in one of these states. Depending to the position of energy state located in band gap, the ones near the conduction band called as “trap” (or “electron trap”) and the ones near the valence band called as “luminescence center” (or “recombination center”) (Krbetschek et al., 1997).

The transition of electron should be radiative in order for the recombination to result with luminescence. Three possible ways of radiative recombination transitions are band to band, band to center and center to center transitions (McKeever, 1985). When an electron directly goes from the excited state (conduction band) to the ground state (valence band) and as a result of this process emission of photon is observed, it is called band to band transition. On the other hand, if there is an indirect recombination process which involves photon interaction

and phonon transfer, these transition are called band to center or center to center transitions and can be both radiative and non-radiative (McKeever, 1985).

Luminescence production due to recombination of electrons and holes may originate from various sources. In Table 2.1, different luminescence types are given with excitation sources.

**Table 2.1:** Luminescence types and excitation sources.

| **Luminescence Type** | **Excitation Source** |
|---|---|
| Photoluminescence | Photons |
| Radioluminescence | Ionizing Radiation |
| Chemiluminescence | Particles produced in chemical reactions |
| Electroluminescence | Electric Current |
| Cathodoluminescence | Electron Beams |
| Mechanoluminescence | Any mechanical action on a solid |
| Thermoluminescence | Heating (after irradiation) |
| Optically Stimulated Luminescence | Photons (after irradiation) |

As seen from the table, different types of luminescence got their names mostly from their excitation source. When emission is generated as a result of absorption of light, it is called to be photoluminescence. If luminescence occurs due to the ionizing radiation ($\alpha$, $\beta$ particles or $\gamma$ rays), it is called radioluminescence. Luminescence is observed in various other ways like as a result of chemical reaction, mechanical action or electrical current. Bioluminescence (which is luminescence emission from a living organism) can be sub-categorized under chemiluminescence due to the fact that emission occurs after chemical reaction in an organism. Thermoluminescence is observed due to heating of a crystalline material and it is not related with the black-body radiation. However, in order for

thermoluminescence to occur, material should be exposed to energy (generally ionizing radiation) before heating. As a result of this energy, electrons go to upper energy levels typically created by crystalline defects and stay at these levels until thermal stimulation occurs. Another type, optically stimulated luminescence, is a similar phenomenon. Even though it can be sub-categorized under photoluminescence (due to excitation source being photons), irradiation of the solid is necessary for optically stimulated luminescence to be observed. In the next section, this type is discussed in detail.

## 2.2 Optically Stimulated Luminesce (OSL)

When an insulator or semiconductor material absorbs energy due to ionizing radiation, free electrons and holes are excited to electronic trapping states mentioned above. The material (generally in crystal form) is put in a metastable state as a result of this excitation unless there is a stimulation. A stimulation (either by light or heat) which liberates charge carries of one sign (i.e. electron), may result recombination of them with the opposite signs of carriers (i.e. holes). As a result of radiative relaxation, luminescence is emitted (Bøtter-Jensen et al., 2003). If the metastable condition of the system is disturbed by heating, the whole process is called Thermoluminescence (TL). If disturbance is caused by light, it is called Optically Stimulated Luminescence (OSL) or Photo-Stimulated Luminescence (PSL).

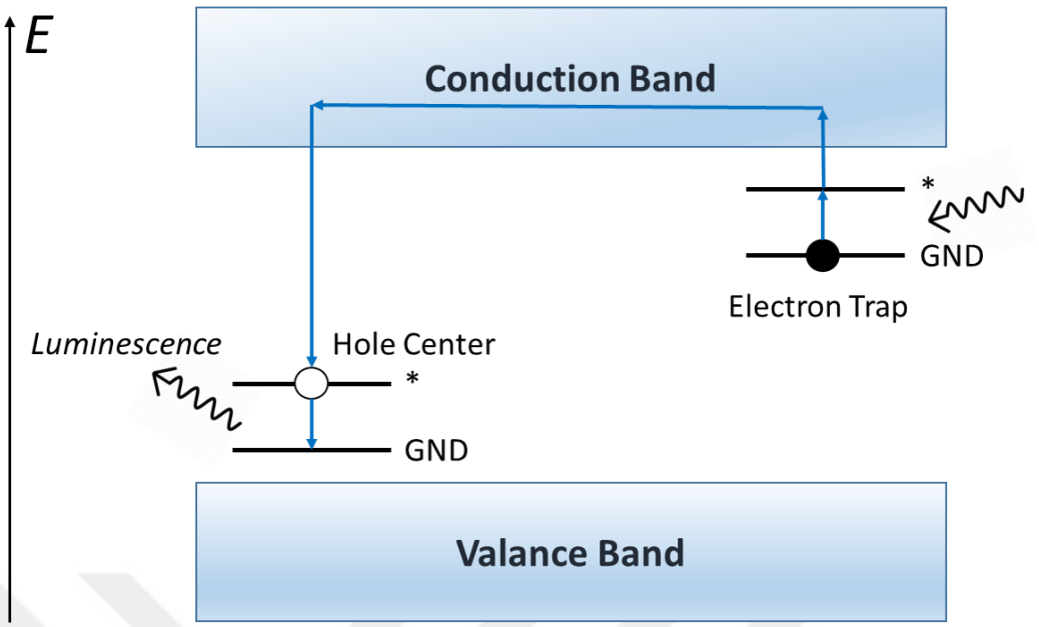



**Figure 2.2:** Energy band model description of OSL phenomenon.

In figure 2.2, energy band model description of OSL is given. Stimulation of trapped electrons by light (UV, visible or infra-red) causes them to move to recombination (hole) center. Radiative relaxation after recombination results in OSL emission.

### 2.2.1 Model and Rate Equations

A series of nonlinear, coupled rate equations are needed to describe the complex processes behind the OSL. During the last century, several models were introduced so as to explain the underlying physical process of stimulated luminescence. The simplest one is called one trap/one center model describing a system that consists of one type of electron trap and one type of hole center. When this model taken into account, charge neutrality for the system can be written as:

$$n_c + n = m_v + m \tag{2.1}$$

where concentration of electrons in the conduction band denoted as $n_c$, concentration of electrons in the electron traps denoted as $n$, concentration of holes in the valence band denoted as $m_v$ and concentration of holes in the hole traps denoted as $m$. During the optical stimulation process, there is no transition to valence band. Therefore, eqn. (2.1) can be written as:

$$n_c + n = m \tag{2.2}$$

By neutrality condition, one may write the rate of change of electron and hole concentrations as:

$$\frac{dn_c}{dt} + \frac{dn}{dt} = \frac{dm}{dt} \tag{2.3}$$

For an arbitrary density of states function $N(E)$ with the energy of $E$, McKeever and Chen (1997) wrote a series of rate equations describing the moving electrons into or out of the delocalized bands during stimulation as follows:

$$\frac{dn_c}{dt} = \int_{E_{Dn}}^{E_c} p(E)\, N(E) f(E) dE - n_c v \int_{E_{Dn}}^{E_c} \sigma_r(E)\, N(E)\big(1 - f(E)\big) dE - n_c \int_{E_{Dp}}^{E_F} \sigma_m(E)\, N(E)(1 - f(E)) dE \tag{2.4}$$

where $f(E)$ is Fermi-Dirac distribution function, $p(E)$ is the probability for stimulation from the trap, $v$ is the free electron thermal velocity, $\sigma_r(E)$ is the capture cross-section for the re-trapping of free electrons and $\sigma_m(E)$ is the recombination

cross-section for free electrons in the conduction band. Here, $E_c$ represents the lowest energy level in the conduction band, $E_{Dn}$ and $E_{Dp}$ represent energy level where the probability of recombination is equal to that of excitation for electrons and holes respectively. They have been dubbed as "demarcation levels" (McKeever and Chen, 1997). The probability for optical stimulation from the trap can be defined as a product of the incident photon flux $\phi$ and the photo-ionisation cross-section $\sigma$.

$$p(E) = \phi(E)\sigma(E) \tag{2.5}$$

Here, it should be noted that $\sigma$ is proportional to absorption coefficient $\alpha$ for optical transitions from a trap. $\alpha$ is a function of stimulation energy $hc/\lambda$, where $h$ is the Planck constant, $c$ is the speed of light in vacuum and $\lambda$ is the wavelength of the stimulating photon. Therefore, it is clear that photo-ionization cross-section $\sigma$ has a wavelength dependence. For thermal stimulations the escape probability can be written as follows:

$$p(E) = se^{-E/k_bT} \tag{2.6}$$

Where $s$ is the attempt-to-escape frequency of trapped electrons, $k_b$is Boltzmann constant and $T$ is the temperature.

Considering only two types of single valued localized states (one electron trap and one recombination center), some simplifications can be made. Setting $N(E)$ as $n\delta(E - E')$ for the trap state in this system, concentration of trapped electron and empty traps becomes

$$\int_{E_{Dn}}^{E_c} N(E')\,f(E')dE' \rightarrow \boldsymbol{n} \quad \text{and} \quad \int_{E_{Dn}}^{E_c} N(E')\,(1 - f(E'))dE' \rightarrow \boldsymbol{N - n} \tag{2.7}$$

and concentration of available hole states for recombination are

$$\int_{E_{Dp}}^{E_F} N(E')\,(1 - f(E'))dE' \rightarrow \boldsymbol{m} \tag{2.8}$$

Using these simplifications and acknowledging that both electron trap energy and recombination center energy are assumed to be single valued, then $\sigma_r$ and $\sigma_m$ are single valued; rate equation (2.4) can be written as follows:

$$\frac{dn_c}{dt} = np - n_c v\sigma_r(N - n) - n_c v\sigma_m m \tag{2.9}$$

Using equation (2.3), The rate of change of electron and hole concentration in the traps may be expressed as:

$$\frac{dn}{dt} = n_c A_n(N - n) - np \tag{2.10}$$

and

$$\frac{dm}{dt} = n_c A_m m = \frac{n_c}{\tau} \tag{2.11}$$

$A_n = v\sigma_r$ is the re-trapping probability (in units of $m^3s^{-1}$), $A_m = v\sigma_m$ is the recombination probability (in units of $m^3s^{-1}$), $N$ is the total available concentration of electron traps (in units of $m^{-3}$), $\tau = 1/A_m m$ is the free electron recombination time (in units of s). Introducing quasi-stationary population of free electron in the conduction band in a way that rate of change of electrons in conduction band is

much smaller than the rate of change of electron and holes in the traps as in equation (2.12).

$$\frac{dn_c}{dt} \ll \frac{dn}{dt}, \frac{dm}{dt} \quad \text{and} \quad n_c \ll n, m \tag{2.12}$$

Assuming the rate of change of electron concentration in the conduction band is negligible, the following can be written:

$$\frac{dn}{dt} \cong \frac{dm}{dt} \tag{2.13}$$

From equation (2.9), $n_c$ can be written as;

$$n_c = \frac{np}{(N-n)A_n + mA_m} \tag{2.14}$$

By plugging $n_c$ in equation (2.14) to equation (2.11), negative of the rate of change of hole concentration equivalently OSL intensity can be given as follows:

$$I_{OSL} = -\frac{dn}{dt} = -\frac{dm}{dt} = \frac{np}{(N-n)A_n + mA_m} mA_m \tag{2.15}$$

However, equation (2.15) cannot be solved analytically. For this reason, some assumptions are to be made. Supposing that that re-trapping probability is very low

and can be neglected$\{(N-n)A_n \ll mA_m\}$, first-order kinetics equation for the OSL intensity may be written as:

$$I_{OSL} = -\frac{dn}{dt} = -\frac{dm}{dt} = np \tag{2.16}$$

And solving this equation by setting $n(t=0) = n_0$, equation (2.16) becomes:

$$I_{OSL} = n_0 p e^{-pt} = I_0 e^{-pt} \tag{2.17}$$

where $n_0$ and $I_0$ are the initial electron concentration in the traps and OSL intensity at $t = 0$, respectively. Hence, this first-order model leads to an exponential decaying OSL intensity as the constant stimulation is applied to the sample. When all the traps are depleted, the OSL signal becomes zero (McKeever et al., 1997).

Even though it is possible to observe such experimental decay curves in application, there are various curve shapes which do not agree with equation (2.17). If re-trapping during stimulation is considered and $mA_m \ll (N-n)A_n$ assumption is made, OSL intensity given in equation (2.15) becomes as follow:

$$I_{OSL} = -\frac{dm}{dt} = -\frac{npmA_m}{(N-n)A_n} \tag{2.18}$$

It can be assumed that saturation of the trap is very unlikely $\{N \gg n\ ,\ R = \frac{A_n}{A_m} \gg \frac{n}{(N-n)}\}$. Recalling the assumption of electron concentration in the conduction band being quasi-stationary and using the equation (2.2), the condition $m = n$ can be written. Then, OSL intensity for second order kinetics model becomes,

$$I_{OSL} = -\frac{dn}{dt} = -\frac{n^2 p A_m}{N A_n} \tag{2.19}$$

The solution of this equation for $n$ is (assuming $n(t=0) = n_0$)

$$n = \frac{n_0 N R}{n_0 p t - N R} \tag{2.20}$$

And substituting equation (2.20) back to equation (2.19), OSL intensity can be written as

$$I_{OSL} = \frac{n_0^2 N^2 R^2}{(n_0 p t - N R)^2} \frac{p A_m}{n A_n} = I_0 (1 - \frac{n_0 p t}{N R})^{-2} \tag{2.21}$$

where $I_0 = \frac{n_0^2 p A_m}{N A_n}$. More complex analysis of OSL kinetics were also conducted by taking additional traps (radiative and non-radiative) and/or recombination centers into account. (Bøtter-Jensen et al., 2003; McKeever et al., 1997; McKeever and Chen, 1997; Whitley and McKeever, 2000; Chen and Pagonis, 2011)

### 2.2.2 Stimulation Modes of OSL

As mentioned before, OSL measurement can be defined as the stimulation of a previously irradiated material (sample) using light source with a determined wavelength (or a range of wavelengths), and observing the emission from the material at a different wavelength using a detector. A stimulation light source and a light transducer are two fundamental elements for OSL measurements (see figure 2.3). In order to prevent scattered stimulation light reaching the detector, strict filtering is very essential for the measurements; since there is a huge difference between stimulation intensity (around $10^2$ mW) and luminescence intensity (in nW

or pW regime) in terms of optical power. More discussion about filtering and OSL measurement setup will be done in Chapter 3.

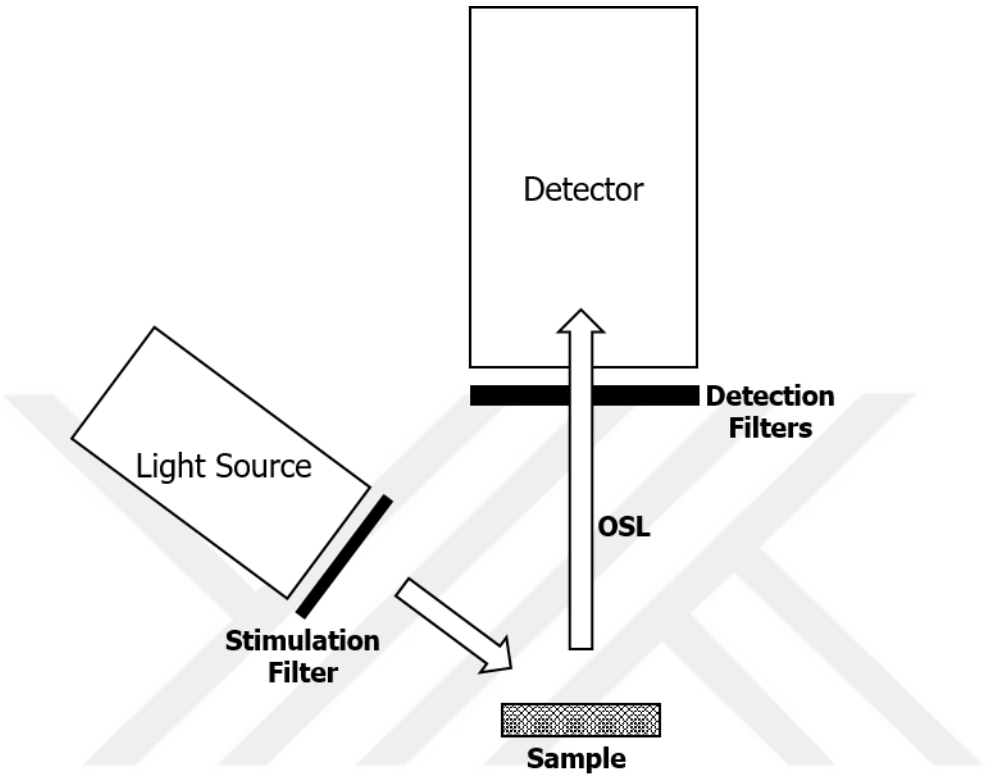



**Figure 2.3:** Schematics of an OSL measurement system with the fundamental elements.

One can apply various of stimulation modes in order to conduct an investigation on a material (i.e. by changing illumination intensity or wavelength of stimulation light). However, there are three commonly applied OSL stimulation modes (see Figure 2.4). If OSL stimulation is done with a steady illumination intensity during the measurement process, the luminescence emission is called Continuous Wave OSL (CW-OSL). When the stimulation source has a linearly increasing illumination intensity with respect to time, the luminescence emission is called Linearly Modulated OSL (LM-OSL). If the stimulation of the material is

applied with pulses, the luminescence emission is called Pulsed OSL (POSL). (see e.g. Bøtter-Jensen et al., 2003)

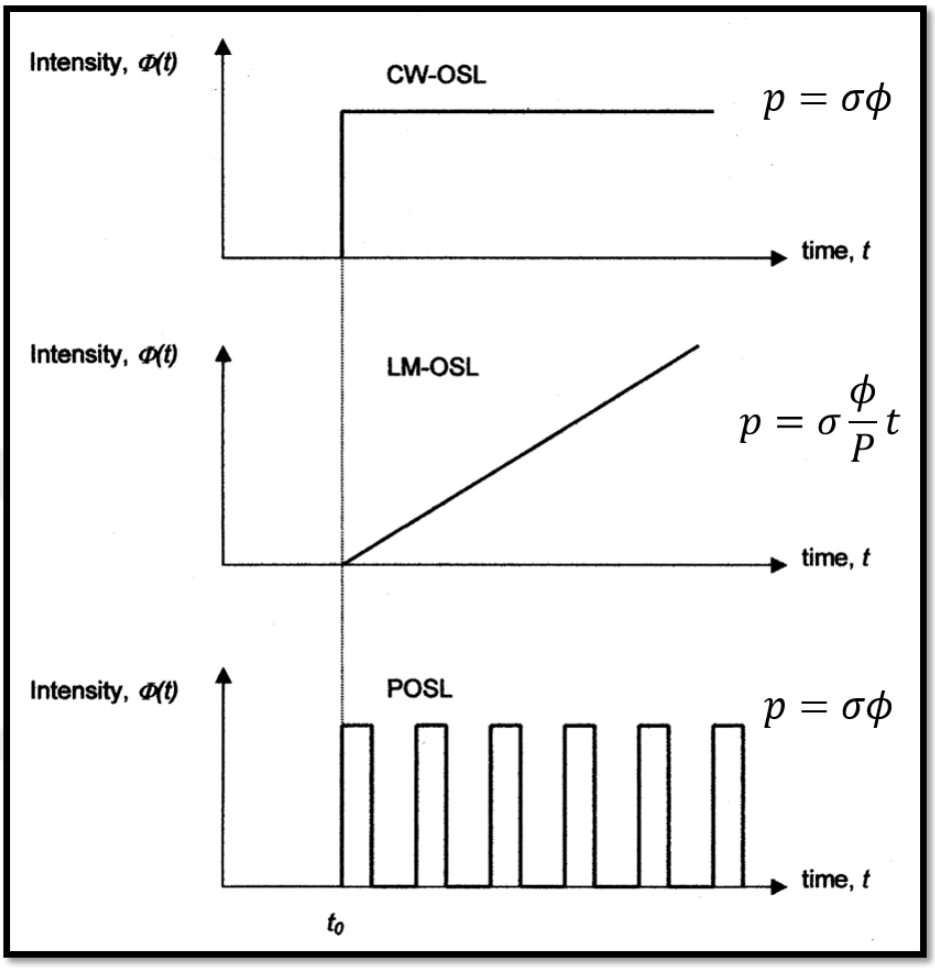



**Figure 2.4:** A summary of three main OSL stimulation modes, namely CW-OSL, LM-OSL and POSL (Bøtter-Jensen et al., 2003).

#### 2.2.2.1 Continuous Wave Optically Stimulated Luminescence (CW-OSL)

CW-OSL mode is the most common way to record the luminescence emission. During the measurement procedure, the stimulation light wavelength and intensity is kept constant since OSL intensity is dependent to these parameters.

While stimulation is done, the luminescence emission is being monitored continuously. Fig. 2.5 illustrates a monotonic decay pattern from the CW-OSL measurement of a BeO (ceramic) sample.

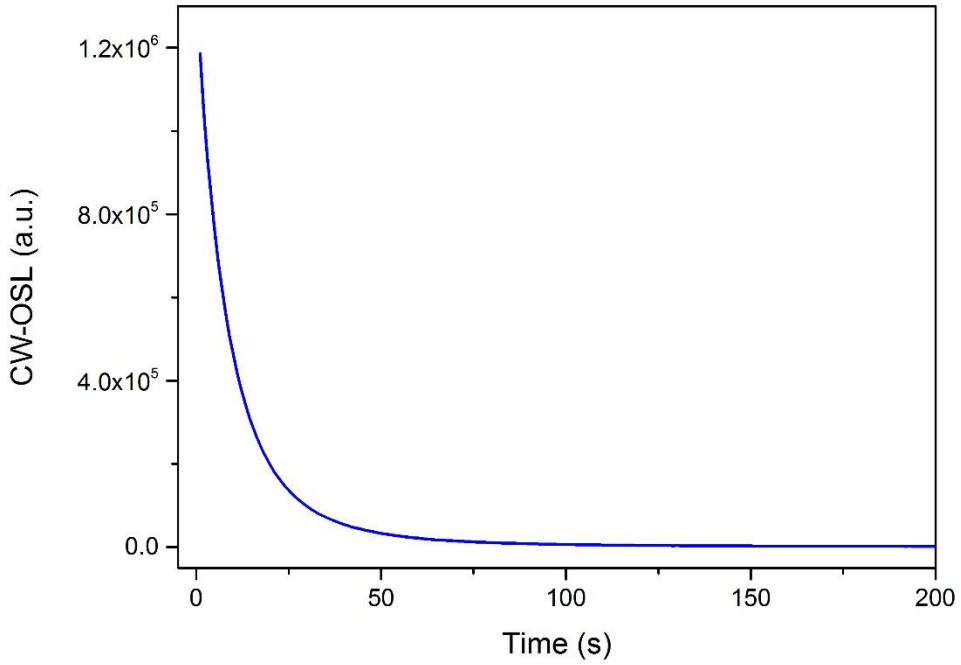



**Figure 2.5:** An example of a CW-OSL decay (irradiated BeO).

In most of the cases for CW-OSL measurement, visible or infrared light is used for stimulation where emission is to be observed at near-UV region. Thus, it is a must to separate the stimulation and emission wavelength.

The CW-OSL decay pattern can be observed either in simple or in multiple decay form assuming there exists more than one type of electron traps. In both cases CW-OSL intensity can be stated as

$$I_{CW-OSL}(t) = \sum_{i=1}^{k} I_{0i} e^{-b_i t} + B \tag{2.22}$$

where for the $i^{th}$ component, $I_{0i}$ is the OSL intensity , $b_i = \sigma_i \phi$ is detrapping probability of electrons. For single exponential form, taking $k = 1$ gives equation (2.13). A stretched-exponential function for the analysis of non-exponential CW-OSL decay is also proposed by Chen and Leung (2002).

#### 2.2.2.2 Linearly Modulated Optically Stimulated Luminescence (LM-OSL)

LM-OSL is called to be produced when the stimulation light intensity is linearly increased with respect to time during measurement process. The technique is introduced by Bulur (1996) as an alternative approach for measuring OSL. When using this stimulation mode, decay components of CW-OSL signal turn into a series of peaks. Each peak is considered as originating from a different type of trap with distinctive photoionization cross-sections. Due to the fact that the traps with large photoionization cross-section empty more quickly than the traps with small photoionization cross-section, it is possible to separate overlapping OSL signals with this technique. For the 1st order kinetics, the luminescence intensity in this mode is obtained as

$$I_{LM-OSL}(t) = Ab\frac{t}{P}\exp\left(-\frac{bt^2}{2P}\right) \tag{2.23}$$

where $A$ is amplitude proportional to both trap population $n_0$, $b$ is proportional to the photoionization cross-section $\sigma$ and maximum stimulation intensity $\phi_0 (b = \sigma\phi_0)$ and $P$ is the total observation time.

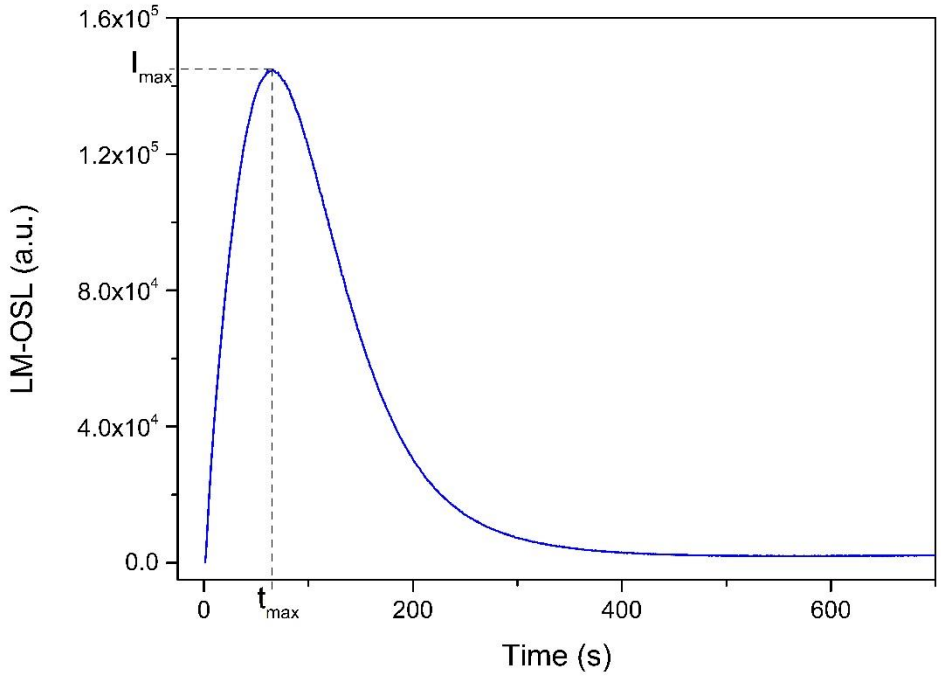



**Figure 2.6:** An example of a LM-OSL ($Al_2O_3$:C).

An example of LM-OSL curve obtained from irradiated $Al_2O_3$:C is given in the figure 2.6. Detailed mathematical description of the LM-OSL curve shape can be found in the paper by Bulur (1996). The application of this mode has been presented in various studies with dosimetric materials such as quartz, $Al_2O_3$:C, BeO, NaCl, $ZrO_2$, ZnS and SrS based phosphors, feldspars. (see Yukihara and McKeveer, 2011 for a review of published studies).

### 2.2.2.3 Pulsed Optically Stimulated Luminescence (POSL)

POSL is another main stimulation mode which was first used by Sanderson and Clark (1994) with the use of pulsed dye laser in the measurement on alkali feldspars. The technique was also used on crystalline $Al_2O_3$:C shortly after its

introduction (Markey et al., 1995; McKeever et al., 1996; Akselrod and McKeever, 1999). Unlike the other two main modes, stimulation is made by brief pulses of light. As a result, it is possible to separate luminescence signal during stimulation and after stimulation in time. Since it is possible to change both pulse width and intensity of stimulation light, absorbed energy per pulse can be controlled.

One of the advantages of using POSL mode is that it is possible to observe luminescence with a lost a negligible amount of dose information. For instance, it is usually possible to redo measurement on a dosimeter and get absorbed dose information. Another one is that signal to background ratio is high when observing luminescence decay. Because signal monitored after the pulse does not contain scattered light coming from the stimulation source.

A type of usage of pulsed stimulation is known as Time Resolved Optically Stimulated Luminescence (TR-OSL). This method only differs from POSL in recording the luminescence. The luminescence signal is resolved in time during measurement of TR-OSL. Hence, this method enables a probable application of analyzing luminescence life times for the recombination center characterization.

Chithambo and Galloway (2000) suggested a simple model for TR-OSL. The rate of change of the number of stimulated electrons $n$ with respect to time is given as follows.

$$\frac{dn}{dt} = pn_0 - \frac{n}{\tau} \tag{2.24}$$

where $n_0$ is initial trapped electron population, $p$ is the stimulation probability per unit time and recombination life time $\tau$. The solution for $n$ at a time $t$ gives

$$n(t) = pn_0\tau\left(1 - e^{-\frac{t}{\tau}}\right) \tag{2.25}$$

and radiative decay of stimulated electrons can be expresses as follows:

$$dI(t) = \frac{n(t)}{\tau} dt \tag{2.25}$$

and luminescence intensity at a given time during stimulation becomes

$$I(t) = I_o \left(1 - e^{-\frac{t}{\tau}}\right) + B \tag{2.26}$$

where $I_o$ is the maximum luminescence intensity in the time period during the stimulation and $B$ is the background signal. Luminescence intensity at a given time after stimulation can be written as follows.

$$I(t) = I_o e^{-\frac{t}{\tau}} + B \tag{2.27}$$

where $I_0$ is the maximum luminescence intensity after stimulation time period.

An example of TR-OSL curve obtained from irradiated BeO is given in the figure 2.7. More information about theoretical considerations of TR-OSL and a comparison between TR-OSL and CW-OSL can be found in the paper by Chithambo (2007).

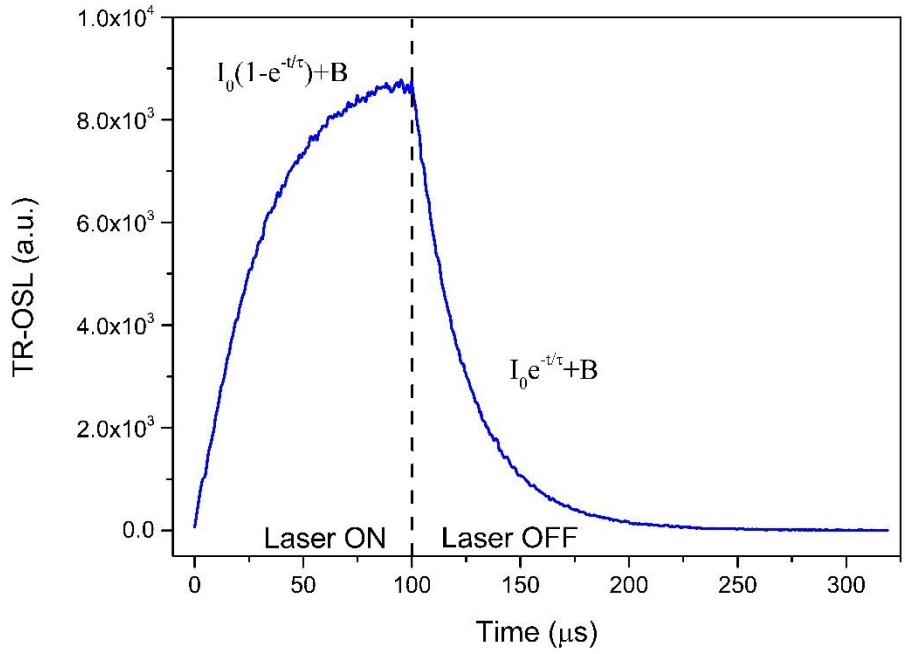



**Figure 2.7:** An example of TR-OSL curve (BeO).

## 2.3 OSL Dosimetry

OSL technique has been widely used for dosimetry applications more than two decades. These applications include personal, medical, environmental, space and, dating or retrospective accident dosimetry. Evaluation of dose absorbed by the matter can be estimated using its relation to OSL signal. As the absorbed dose increases, there is an enhancement in the OSL signal coming from the material.

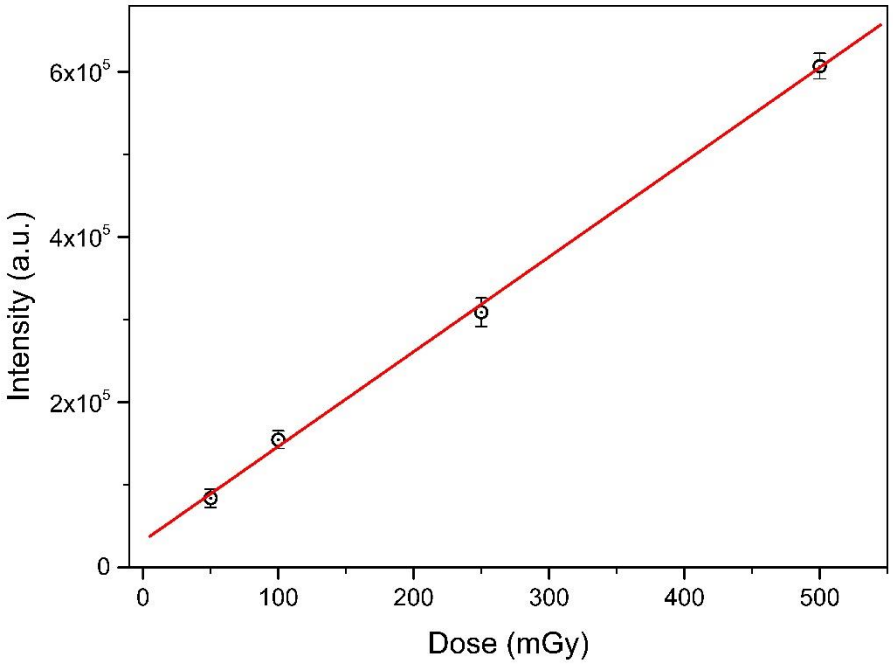



**Figure 2.8:** An example of dose response (BeO).

In order to obtain absorbed dose information using OSL dosimetry, one should first calibrate the dosimeter using a known radiation source. The OSL signal with respect to known absorbed dose should be gathered for that specific dosimeter. Then, OSL decay curves of various doses should be recorded. Using these curves, one should calculate OSL dose response of that material. In order to do that, different approaches such as deconvolution by curve-fitting, integration of curve or summing first several data of the curve may be used. As a result, OSL dose response information for varying doses will be obtained. One can find a dose response trend of a dosimetry by fitting these data points to a line or a curve. In figure 2.8, dose response data of BeO is given together with its linear fit in the range from 50 mGy to 500 mGy as an example.

Only after having a dose response trend curve/line of the material, OSL signal can refer the dose absorption. Similar approach is used for archeological

dating (see Huntley et al., 1985 and Hütt et al., 1988, Wintle and Adamiec, 2017). The studies reported by Afouxenidis et al. (2007), Zacharias et al. (2007), Yukihara et al. (2010), Yukihara et al. (2014), Nascimento and Hornos (2010) can be referred for more detailed information about dosimetry applications.

There are some advantages of OSL dosimetry technique over TL for personal dosimetry monitoring. One of them is that one can adjust the performance of the dosimetry just by changing the power of the stimulation light. By doing that, sensitivity of both low-dose and high-dose end of the dose response can be increased. Another one is that the ability to re-read the OSL signal, accordingly measure exposed dose, by adjusting the stimulation power. Even though, some information stored in dosimeter is lost (some charges were depleted), dose estimation still can be done.

# CHAPTER 3

# SCIENTIFIC INSTRUMENTATION

## 3.1 Instrumentation Basics

In order to understand diverse phenomena of nature, space, or man-made objects; the comprehension of the state, amount, or value of different elements is necessary in the scientific and technological world. Obtaining the knowledge of the state, amount, or values of different elements is called as measurement (Bhuyan, 2011). It is certain that measuring instruments have a significant role in the physical sciences and engineering. A group of systems which enables measurement and maintaining retroactive control of the measurement process is termed as instrumentation (Placko, 2007).

In this sense, an instrumentation system can be divided into two sub-systems: measurement and control system. A measurement system consists of instruments which enable acquiring information and data concerning tested material or object. Generally, a control system consists of elements and instruments that enables implementation of control using a feedback process. There are also open loop systems as well. A diagram of measurement and control system is given in figure 3.1. A measurable quantity, which is measured as *X(t)*, is transmitted by a signal *M(t)* at the input of the measurement chain. After that this signal is characterized by a transfer function *T(t)* and measurement chain crates an exit signal *S(t)* which has a relationship to the input quantity by *T(t).* In other words, an instrumentation system makes measurements and gives the user a numerical output value corresponding to the variable being measured (Bolton, 2015). The whole system can be finalized by

a feedback loop with a transfer function *B(t)*. This transfer function carries out the control parameters of the object. Physically, measurement chains consist of devices that converts one form of energy to another. They are called as transducers. If the exit signal of the transducer is electrical, that means the measurement chain is a sensor (Placko, 2007).

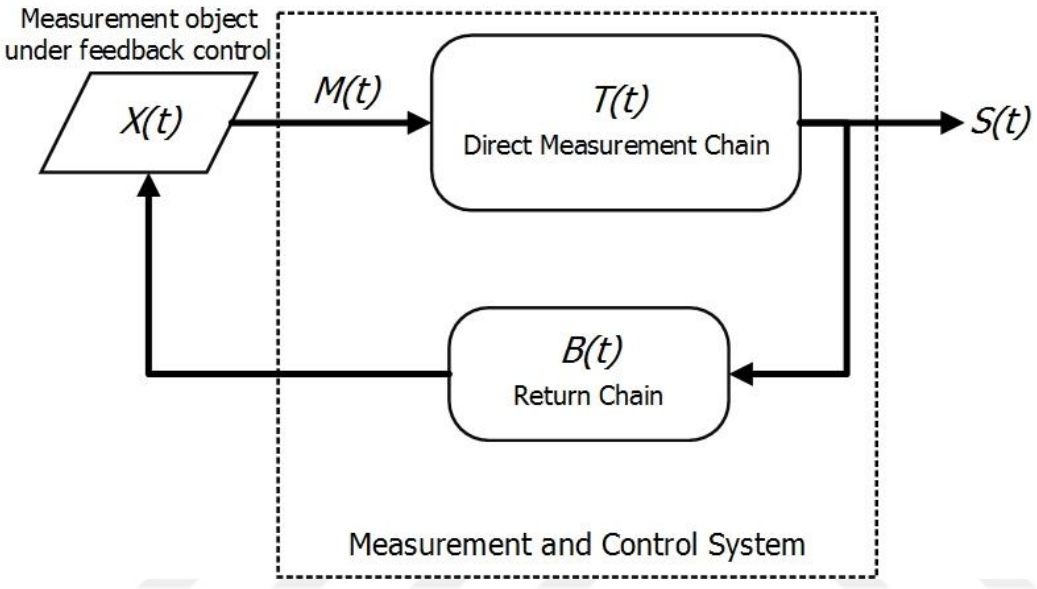



**Figure 3.1:** Measurement and Control System.

Accuracy, precision and reliability of a system have importance for scientific data and results. (McMahon, 2008). Therefore, choosing suitable tools and devices for the systems enhances the quality of instrumentation. There are three functional elements that forms an instrumentation system for making measurements: sensor, signal processor and data presentation.

Sensor (or detector) is the element which is constantly in contact with the measurement process. A variable is being measured and an output is given to the rest of the measurement system by this element. A thermocouple (which has a temperature input and electromotive force output) is an example for a sensor, when it is combined with measurement electronics.

Signal Processor is the element which gathers the output information from the sensor and make a suitable conversion for display or for oncoming transmission in some control systems. Active and passive analog filters, digital filters and amplifiers can be given as an example for signal processor.

Data presentation element enables the user of the measurement device to observe the measured value in a recognizable format. In other words, it converts the signal coming from the system into observable output form. A liquid crystal display screen, a computer user interface software or a galvanometer can be given as examples of this element.

Figure 3.2 illustrates how three basic elements (sensor, signal processor, data presentation) comprises a measurement system.

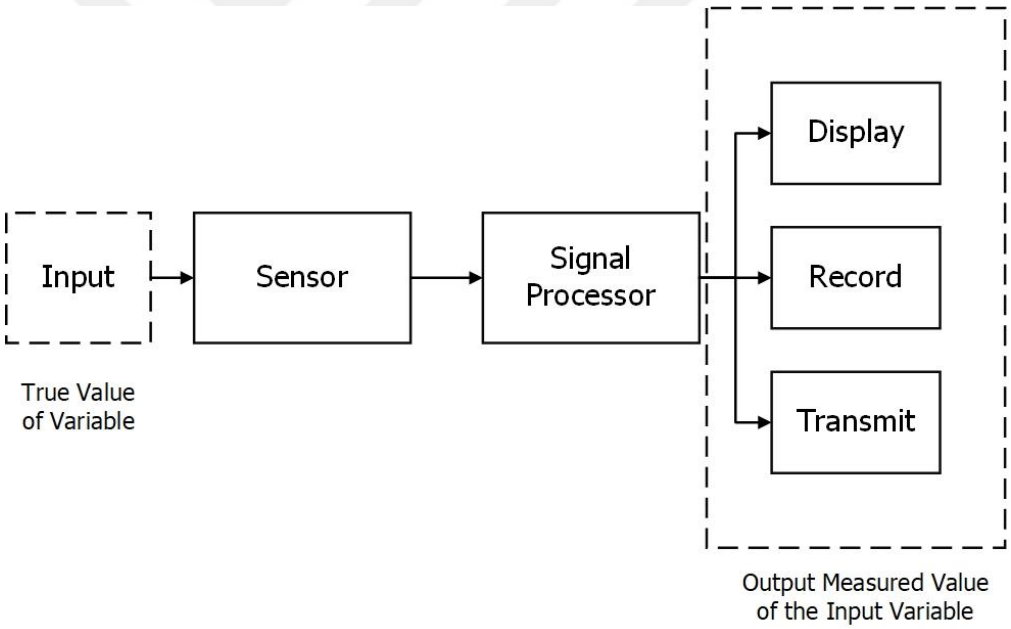



**Figure 3.2:** A diagram of measurement system elements.

Modern implementation of measurement (or control) systems are based on microcontrollers. It is advantageous for instrumentations to have multiple inputs for sensors, ability to give output through different channels, signal and data processing

capability (analog and digital processing, application of mathematical methods and calculations) together with on board memory, ability to control other instruments in one device.

### 3.1.1 Characteristics of an Instrumentation System

Performance of instrumentation systems and functional elements is another point to take into consideration. There are several terms that determine the performance characteristics of a system. These terms can be listed as Resolution, Accuracy, Error, Range, Precision, Repeatability, Reproduction, Sensitivity and Stability.

The smallest amount of an input signal (measurand) which can be reliably detected with certainty by a measurement device or instrument is called to be *Resolution* of the system. *Accuracy*, is another characteristic of an instrumentation system, defines the proximity between measured value and the actual value of the process variable being measured. In other words, if the measured value is close the actual value, the system is called to be more accurate. Percentage of full-scale deflection (FSD) representation is a common approach in order to express the maximum difference between the actual value and the measured values (Sheel, 2014). *Error* is a term which defines the difference between the true value measured quantity and the result of the measurement. It can be stated as follow

$$Error = Measured\ Value - True\ Value$$

Errors may be originated in number of ways. Bolton (2015) can be referred for detailed information about some of the errors that are come across in specifications of instrumentation systems. The *range* of a variable is determined by its minimum and maximum obtainable values.

In order to characterize the degree of freedom of an instrumentation system, *precision* term is used. If a measurement instrument gives a small spread of readings, it is called a high precision instrument. If it gives a large spread of reading, it is called a low precision instrument. *Repeatability* and *reproducibility* are the terms

related to the instrument's precision. When a system has an ability to give the same output for repeated measurements (random fluctuations in the environment is low), the measurement is called as repeatable. When a system has an ability to give the same output after disconnecting it from a constant input and reinstalling, it is called as reproducible.

The *sensitivity*, which is another characteristics of a measurement system, is determined by the ratio of output to input. In other words, it designates how much the output of a measurement system changes when the measured quantity changes by a given amount (Bolton, 2015).

*Stability* is another important characteristics of instrumentation. It is expected that a measurement device to give the same output when it is used to measure a constant input over a period of a time.

Besides the characteristics mentioned above, there are also dynamic characteristics which refers to behavior of an instrument related to time. One of the commonly used dynamic characteristics is called *response time*. It is difference between the time when input is recognized by the system and the time when the system gives an output corresponding to a specified percentage of input value (in general 90%). Another dynamic characteristic is *rise time*. It is the time elapses when the output signal rises from a specified percentage (i.e. 10%) to another specified percentage (i.e. 90%) of a steady-state output. *Fall time*, which is defined as time elapses during the fall of the output signal to specified percentage, may also be significant for some systems.

## 3.2 Optically Stimulated Luminescence Measurement Technology

Some significant characteristics of instrumentation and measurement systems and the developments and significance of automated readers was mentioned previously. In addition to these, there are other points to take into consideration when designing an OSL measurement system.

In figure 3.4, a simplified diagram for OSL measurement instrumentation is given. In general, system consists of a measurement detector, a stimulation source and measurement control electronics. As it can be seen from the figure, stimulation light and luminescence emission is separated using filters. During stimulation, which is controlled by an electronic control unit (such as microcontroller), detected luminescence emission is recorded. In this section, various light detection sensors, different types of stimulation sources are discussed for utilization in luminescence measurement.

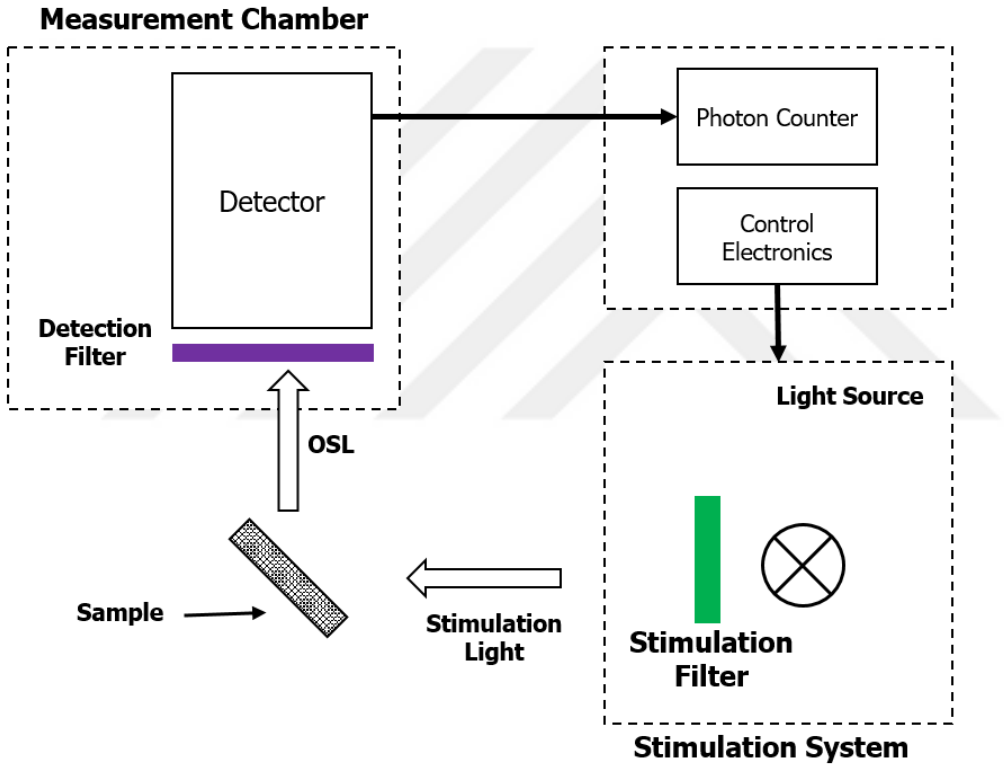



**Figure 3.3:** A simplified diagram of setup for measuring OSL.

### 3.2.1 Light Detection Unit

A light detection unit in a luminescence measurement system is an important transducer which converts the emitted luminescence light to electrical current signal. Light detection unit is going to be discussed under three categories: Photomultiplier tube (PMT), avalanche photodiode (APD) and multichannel detectors.

#### 3.2.1.1 Photomultiplier Tube

A photomultiplier tube (PMT) mainly converts incoming light into amplified electrical signal. This process is based on external photoelectric effect (Gilmore, 2014). A PMT consists of following parts in vacuum case with a window: a photocathode, an array of dynodes, and an anode. A schematic diagram of a PMT is shown in figure 3.4.

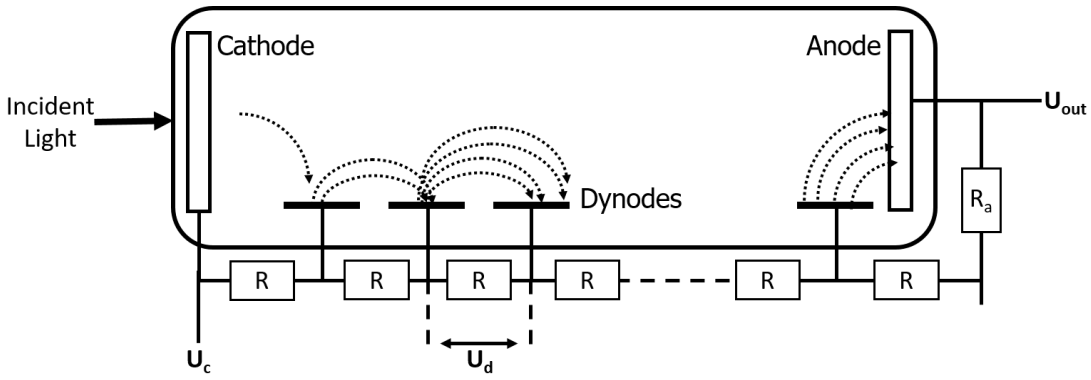



**Figure 3.4:** The schematic diagram of a PMT. $U_c$ is cathode voltage, $U_d$ denotes the potential difference between two dynodes, R is resistor. (Re-drawn after Tkachenko, 2006)

Incident photons passing through the light input window hits photocathode and eject electrons as a result of photoelectric effect (with quantum efficiency less than one). Then, photoelectrons, which are accelerated due to electric field between anode and cathode, hits the first dynode and ejects $\delta$ number of secondary electrons by transferring some of its kinetic energy to them. Here $\delta$ is the coefficient that determines number of electrons released from a dynode. These secondary electrons hit the next dynode and emission of more electrons occurs. This process continues until electrons reach the anode. Hence, a great increase in the number of electrons is observed (around $10^5$-$10^7$). This increase is called as photomultiplier gain. The acceleration of primary and secondary electron is done by applying a high voltage between electrodes. This is around 1000 V in general (Pelant and Valenta, 2016).

Light detection using PMT can be done in two modes. One of them, which is called DC mode, can be operated by taking the signal coming to anode as direct current and record it with respect to time. Only DC portion of PMT output signal is detected with the help of an amplifier and a low pass filter. However, in some experiments, the light intensity is too low to be measured in DC mode. In this case, photon counting mode is used. Using a fast pulse amplifier and a pulse height discriminator, it is possible to directly count the single pulses originating from the photons. It is more likely to have better sensitivity using this mode (Bøtter-Jensen et al., 2003).

Technical parameters of PMT is also significant for an instrumentation system. *Spectral range of sensitivity*, which is the spectral region PMT gives response to electromagnetic radiation, is a significant parameter of PMT. It is determined by the work function of the photocathode material. Another parameter, *electron transit time,* determines the time elapses after the ejection of photoelectron until it reaches the anode. *Noise* or *Dark Current* of a PMT is also another parameter. It is the current due to thermionic emission of electrons from the photocathode and the dynodes.

#### 3.2.1.2 Other Detectors

One of the detectors which can be used for luminescence detection is avalanche photodiode (APD). This detector is based on a reverse-biased p-n junction. As a result of absorption of incident photons, electron-hole pairs is created. The generated charges are separated under the influence of electric field due to p-n junction. Therefore, a voltage difference is observed as output. During the separation of charges, energetic electrons can create new electron-hole pairs by impact ionization. Hence, there occurs multiplication of photo-carriers; in other words, there is an internal gain inside the photodiode. Having small active area limits the usage of these detectors. The sensitivity of APD is less than PMTs, even with a very good focusing of luminescence emission. Moreover, these detectors show wavelength dependency which requires addition normalization after measurement (Gilmore, 2014).

In addition to single channel detectors like PMT and APD, there are detectors used in luminescence detection and designed as one or two dimensional arrays of photosensitive solid-state devices. Photodiode arrays and charge-coupled-device(CCD) cameras are two of the most known types of multichannel detectors. More detailed information regarding working principle of multichannel detectors and usage in luminescence detection can be found in Bøtter-Jensen et al. (2003) and Pelant and Valenta (2016).

### 3.2.2 Stimulation Sources

There has been a great increase in the number of types of light sources since the first application of OSL. As a result of advances in the semiconductor technology, there are Light Emitting Diodes (LED), semiconductor laser diodes in addition to flash lamps. In this section, some of the light sources, which can be suitable for luminescence measurement system, is discussed.

Even though flash lamps, halogen lamps and deuterium arch lamps are now largely replaced by semiconductor light sources, they are still in use for luminescence measurements and spectroscopy. The advantage of using such a source is the wavelength range. Especially, it is hard and expensive to achieve stimulation source in the middle and far ultraviolet (UV) region, flash lamps and deuterium lamps are still in use with strict filtering and preferred by researchers who desire UV excitation of samples.

However, these lamps have some limitation such as repetition rate and pulse width. For this reason, the usage of cavity-pumped dye laser was increased in the 1980s (Gilmore, 2014). They are mainly used for shorter luminescence lifetime measurements since it is possible for a dye laser to provide pulse widths of picoseconds (ps) and repetition rate up to megahertz (MHz). The most common pump lasers are, argon ion laser (emission at 514 nm) and yttrium aluminum garnet (YAG) lasers (doubled frequency emission at 532nm, tripled frequency emission at 355 nm).

After the developments in semiconductor technology, LEDs and Laser Diodes are favored as a stimulation sources for luminescence measurements. Despite the fact that they suffer advantages such as tuning and having emission as a combination of different wavelengths instantaneously, various of emitting sources are available on the market in the range between 250 nm and 1300 nm. Moreover, their repetition rates go up to tens of MHz, as they present pulse widths around 1 ns. Furthermore, LEDs are very cost efficient comparing to the other stimulation sources used in luminescence measurements (Gilmore 2014).

The specific usage of these stimulation sources for OSL measurements are explained and illustrated in book by Bøtter-Jensen et al. (2003).

### 3.3 Open-Source Hardware and Software in Instrumentation

The term *open-source* has been started to be used in the late 80's by several hackers (free and open-source software developers) during a strategy session. Its popularity increased especially after the foundation of Open Source Initiative (OSI) in 1988 (Bretthauer, 2002). Free/open-source software (FOSS) defined as a software that is available in source code form. It is possible to use, study, copy modify and redistribute the source code without restriction (Pearce, 2014).

Open-source term does not only indicate that its source code or blueprint is accessible. There are other criteria that open-source software or hardware should meet such as free redistribution, integrity of the author's source code. The Open Source Definition (OSD), which was originally written by Bruce Perens, describes these ten criteria (see https://opensource.org/docs/osd, Accessed on April 5$^{th}$, 2017). Kavanagh (2004), DiBona (1999) can be referred for more detailed information about open source software.

Scientific hardware designs were also affected by open and collaborative principles of FOSS. As a result, there are publicly available hardware designs to study, modify, distribute, make and sell. They are called as free and open-source hardware (FOSH) (Pearce, 2014).

#### 3.3.1 Microcontrollers

Microcontrollers have a significant role in FOSH designs. Most of the open-source projects involves electronic sensing and control are in need of a microcontroller. Farmbot (https://farmbot.io/, Accessed on April 5$^{th}$, 2017), Project Ara (https://atap.google.com/ara/, Accessed on April 5$^{th}$, 2017), RepRap (http://reprap.org/, Accessed on April 5$^{th}$, 2017) , OpenKnit (http://openknit.org/, Accessed on April 5$^{th}$, 2017), OpenROV (https://www.openrov.com/, Accessed on

April 5th, 2017), APM:Copter (http://ardupilot.org/, Accessed on April 5th, 2017) are some of the projects that utilizes a microcontroller. One of the most successful open-source microcontroller environment is Arduino electronic prototyping platform (http://www.arduino.cc, Accessed on April 5th, 2017).

### 3.3.2 Arduino Development Platform

Arduino, which is a microcontroller-based development platform, is a result of one of aforementioned FOSH projects. With the help of a microcontroller which is mounted on a circuit board, it is possible to program the input and output pins of the microcontroller hardware and make interaction with these pins using a personal computer. In order to upload a program to an Arduino board, the Arduino Integrated Development Environment (IDE) is generally used. Since Arduino is an open-source hardware project; specifications of electronic components, all circuit board designs and the IDE software is freely accessible for anyone to use or make modifications. Hence, it is possible to find inexpensive Arduino-like or Arduino-compatible microcontrollers from private manufacturers all around the world.

Since the first introduction in 2005, usage of Arduino microcontroller boards is greatly increasing. Some examples of Arduino based projects can be found in Karvinen and Karvinen (2011). Due to the ease of use, low cost and standardized components, Arduino platform also is utilized by researchers in development and implementation of devices for variety of applications (see Bri et al., 2008; Buechley and Eisenberg, 2008; Zhang et. al, 2009; Bergmann et al., 2010; Gordon et al., 2010; Sarik and Kymissis, 2010).

There are seventeen different official Arduino boards available as of today. In addition to these, there are also unofficially designed models. In this study, as a microcontroller of the measurement system, Arduino DUE board is used. This board is based on microcontroller Atmel's AT91SAM3X8E 32-bit programmable microcontroller (Atmel Corporation, San Jose, CA USA). It has 512 kilobytes (kB)

flash memory for all user applications. There exist 54 digital input/output pins in addition to 12 analog input and 2 analog output pins. This board operates with 3.3 V with an oscillation speed 84 MHz. A pinout diagram of Arduino DUE is given in Appendix A (see figure A.1). For more detailed information and specifications of chips, datasheets can be downloaded from Arduino project (http://www.arduino.cc) and Atmel http://www.atmel.com) websites.

#### 3.3.2.1 Software and Communication of Arduino

Arduino IDE (software environment for programming and interacting with board) installation for Windows, Mac OS and GNU/Linux is available online on project website. Advantage of the Arduino IDE is that users can upload codes written in *Arduino* language (simplified version of *C++* for ease of usage of the boards). This simplified version is fundamentally a library written for Arduino development boards. Moreover; since Arduino project is an open-source environment, there are a great number of libraries written by the community for various of purposes. *LiquidCrystal, SoftwareSerial, Stepper, SD, GSM, WiFi, Tone, I2S, Servo* and *Firmata* are some examples of these libraries (see https://www.arduino.cc/en/reference/libraries, Accessed on April 22th, 2017).

There are several methods of communication possible for Arduino with external electronics, sensors and computers in addition to its digital and analog input/output pins. The most known and widely used communication protocol for Arduino is RS-232 (the standard serial communication). Two communication lines are used for this protocol and there exist two pins on Arduino as *Rx* (to receive), *Tx* (to transmit). In more developed models such as Arduino Mega or DUE, there are four different pairs RS-232 communication pins designated for enabling more connections. Moreover, many Arduino models has its own USB-to-serial converter chip embedded on boards. As a result, the boards can create connection with a personal computer (PC) by creating a virtual serial port (Fisher and Gould, 2012).

Another one of the protocols that can be used for communication is The Inter-Integrated Circuit (I2C) (Philips Semiconductors). There are two input/output pins designated for I2C communication. It is possible to connect more than one device to I2C connection as long as each device has its own unique identification number. The other protocol which is available on Arduino boards is the Dallas 1-Wire protocol (Dallas Semiconductor). This protocol uses a single input/output pin. The Serial Peripheral Communication (Motorola), which is known as SPI, also exists on the most of the boards.

### 3.3.3 Python Programming Language

*Python* is one of the popular high-level programming language to write software in different application domains and purposes (Rossum, 1995). *Python* is defined that it is an interpreted, portable, interactive, interfaced, object-oriented, open-sourced, easy to understand and use language (Nelli, 2015).

The language is interpreted, in other words pseudo-compiled. This means that in order to run a code written in *Python*, an interpreter program (interprets the source code and run it) is necessary. Therefore, there is no compile time needed for *Python* code unlike the programming languages like *Java* and *C*.

*Python* is called to be portable due to the fact that installing an interpreter to most of the operating systems (such as GNU/Linux, Windows, Mac OS) is doable while the interpreted code remains the same. Hence, it is utilized as a programming language for small devices like Raspberry Pi (https://www.raspberrypi.org/, Accessed on April 23$^{th}$, 2017). *Python* is also known to be interactive language (Hughes. 2011). Interpreter enables users to execute commands instantly and decide what to write to next line depending on the output. It is possible for *Python* to interface code written in other programming languages as well. Especially for these reasons, *Python* has become popular among scientific community (Nelli, 2015).

In *Python* programming language, specifying classes of object does not require construction unlike *Java* or *C*. *Python* is also an open-source programming language, which means that its reference implementation (known as CPython) is free to modify, distribute.

Ease of use of *Python* language cannot be ignored. Division or categorization of written functions is determined by indented and new lines instead of various types of parenthesis and punctuations line in other popular languages. In addition to that, there is a huge community of this language, resulting various of libraries available for users.

# CHAPTER 4

# DESIGN AND CONSTRUCTION OF THE AUTOMATED OSL READER

Within the framework of this study, an optically stimulated luminescence reader was designed and constructed with an automated sample changer unit for multi-sample measurements. The reader was designed using open-source hardware and software. In this chapter, properties and components of the automated OSL reader will be explained. Measurement chamber, motorized sample changer unit, hardware and software of microcontroller and user interface software discussed in detail in order to make an understanding of design.

## 4.1 Structure and Properties of the Optically Stimulated Luminescence Reader

In order to observe luminescence emission from materials which are previously exposed ionizing radiation, optical stimulation is one of the methods which is commonly used (see Bøtter-Jensen, 2003). It is clear that a stimulation source and a detection unit is necessary in the design of any OSL measurement system. Furthermore, they should be cooperated by a controller and the signal coming from the measurement unit should be recorded with respect to time so that further analysis can be performed. Optionally, a motorized sample changer unit and control of this unit is necessary for automated measurements.

In this study, a measurement chamber carrying both stimulation and detection unit, a sample changer unit which can be loaded up to eight samples is designed. Both measurement process and sample position control are provided and

monitored by an open-hardware design Arduino DUE microcontroller. In addition, a novel measurement control software was written in *Python programming* language. Figure 4.1 demonstrates the simplified block diagram of the automated OSL Reader.

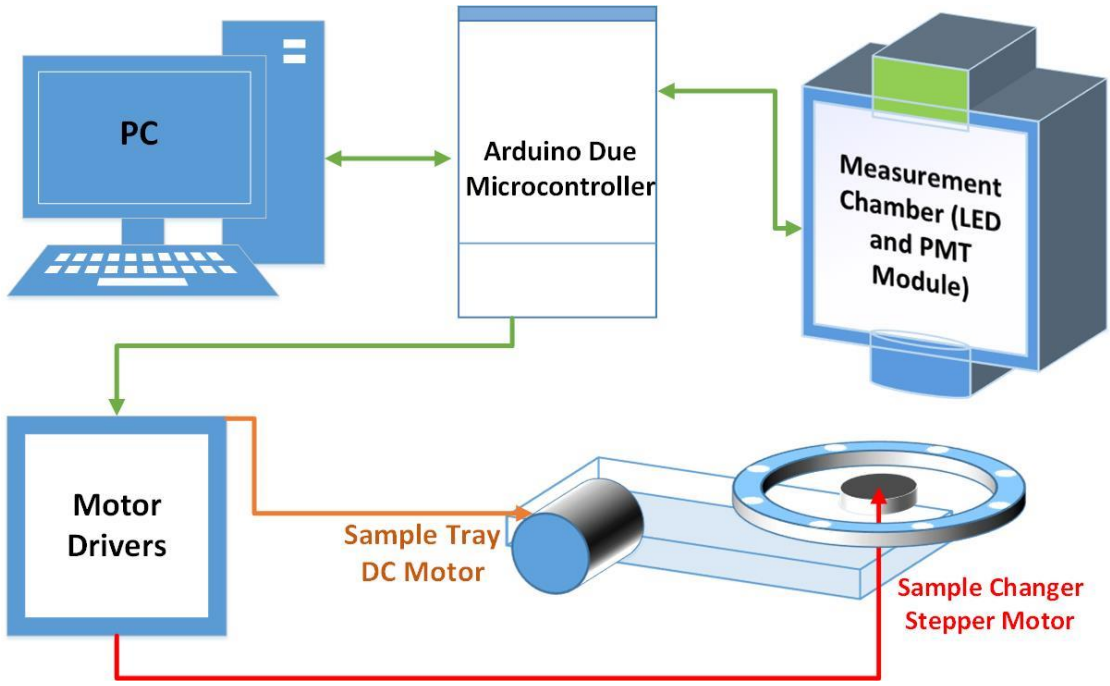



**Figure 4.1:** Simplified block diagram of the automated OSL reader.

As it can be seen from the block diagram, sample changer unit driven by motor drivers is controlled by an Arduino DUE board. In addition, measurement process is also conducted by the microcontroller. There is also constant communication between the measurement control software and the microcontroller which enables the user interaction. All of the aforementioned components and parts will be described in the following section.

The automated reader is able to perform three modes of OSL measurement. These measurements are CW-OSL, LM-OSL and TR-OSL. A user can enter proper measurement parameters for selected measurement mode using a PC. It is possible

to create measurement sequences for consecutive automated measurements using measurement control software.

## 4.2 Components and Parts of the Optically Stimulated Luminescence Reader

The OSL measurement system described in this study comprises of four main parts. These parts are measurement chamber, a motorized sample changer mechanism, electronic control unit and user interface software. Measurement chamber, sample changer tray mechanism, microcontroller, motor drivers and all electronics of the systems are put in a light-tight box with dimensions: 60 cm length, 45 cm width and 35 cm height.

### 4.2.1 Measurement Chamber

The measurement chamber is the main instrument inside the OSL reader that enables luminescence measurements. It consists of a stimulation source, a detection unit, and associated optics for focusing and collecting stimulation light and emitted luminescence. A simplified sketch of the measurement chamber is given in figure 4.2.

As a stimulation source, a high power LED (Cree, Product Code: XQEBLU-SB- 0000-000000Y01) is used. The LED is place inside an aluminum holder which also serves as a heat sink. The emission of the LED used in the reader has a peak at 475 nm with the full width half maximum around 32 nm (Cree XLamp XQ-E LEDs, 2013). In order to avoid the short wavelength emission from the LED, a glass long pass filter with 25 mm diameter and 3 mm thickness is used (Schott GG-420). The stimulation light is collimated using acrylic non-imaging optics.

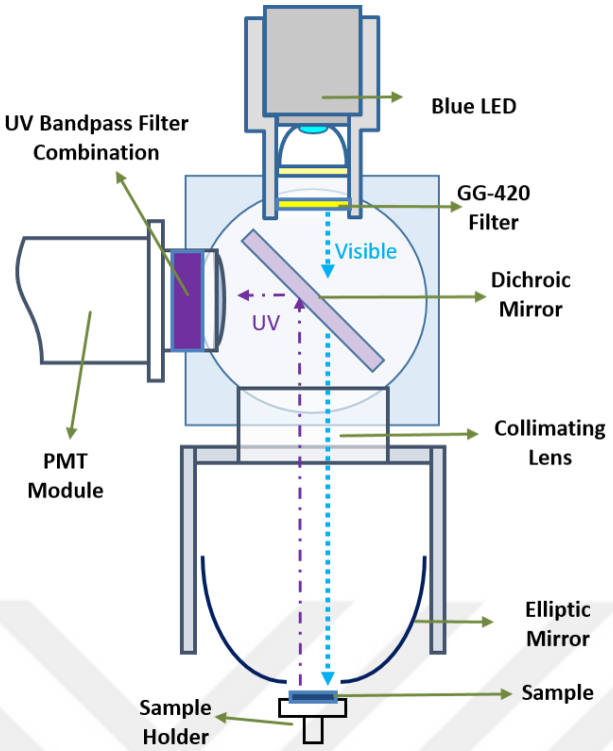



**Figure 4.2:** Simplified sketch of measurement chamber.

A photomultiplier tube (PMT) module (Hamamatsu, Product Code: H7173), which has a built-in high voltage (HV) supply and constant level discriminator, is used as a detection unit. The PMT inside the module has a spectral response between 300 nm and 650 nm (Hamamatsu, R3550). The discriminator inside the module gives transistor–transistor logic (TTL, 0-5V) pulses for photon counting. In order to avoid the PMT to be exposed to visible light emitted from the stimulation source, two ultra-violet (UV) band pass filter with total thickness 5 mm and diameter 25 mm (Hoya U340+Schott DUG 11) is placed in front of the window of the PMT.

Separation of stimulation (visible) and luminescence (UV) lights was achieved using a dichroic mirror (beam splitter) placed at an angle of 45°. This dichroic mirror enables the visible light to transmit whereas reflects UV light. In addition, a UV grade biconvex lens and elliptical mirror is placed under the measurement chamber so that the LED illuminates the sample holder (a diameter of 10mm) uniformly. The effective focal length of the system is approximately 40 mm.

An H-bridge circuit based on L298N driver was used for switching (turning) ON and OFF the stimulation LED.

The measurement chamber is designed and constructed in a way that visible stimulation/near-UV detection OSL measurements can be handled. In other words, the design provides following: the light emitted from the LED transmits through the dichroic mirror, reaches and stimulates the sample. Then, the collimated UV emission of luminescence is reflected from the dichroic mirror reaches the detector. It is also possible change configuration and filters easily if necessary for different configuration. The measurement chamber has dimension of 49 mm length, 40 mm width and 83 mm height. A picture of measurement chamber (figure B.3) was given in Appendix B.

### 4.2.2 Motorized Sample Changer Unit

The automated OSL reader has a sample holder wheel (diameter = 135 mm) made of aluminum. The sample holder's outer rim is machined for holding eight 10 mm stainless steel sample holder cups. It is painted in black to reduce reflection of stimulation light. The sample is positioned under the measurement chamber for the measurement with the assist of a rotary stage (Micos DT-90). The rotary stage is driven by a stepper motor with a precision of 0.015 degrees with the help of a gear mechanism (24000 full steps per revolution). The origin for sample positioning is determined with the help of two sensors. A mechanical switch determines the origin of rotary stage, where an opto-switch determines inside position of tray. There is also another opto-switch for determining the outside position of the tray. The sample wheel is attached on the top of a tray which can move in one dimension on a belt driven rail mechanism. Hence, the sample wheel can be taken out of the light tight box for loading or unloading the samples. The distance between inside and outside position of tray is 48 cm. A top view picture of motorized sample changer unit (figure B.2) can be seen in Appendix B. The movement is provided by a direct

current (DC) motor (Faulhaber Motor, 2342L012CR) mounted to the tray. Open hardware H-bridge circuit (L298, STMicroelectronics, 2000) is used for driving stepper and DC motor. Moreover, there are two buttons placed on the front panel of the OSL measurement device such that it is possible to take tray in/out and change the position of samples without the need of user interface software.

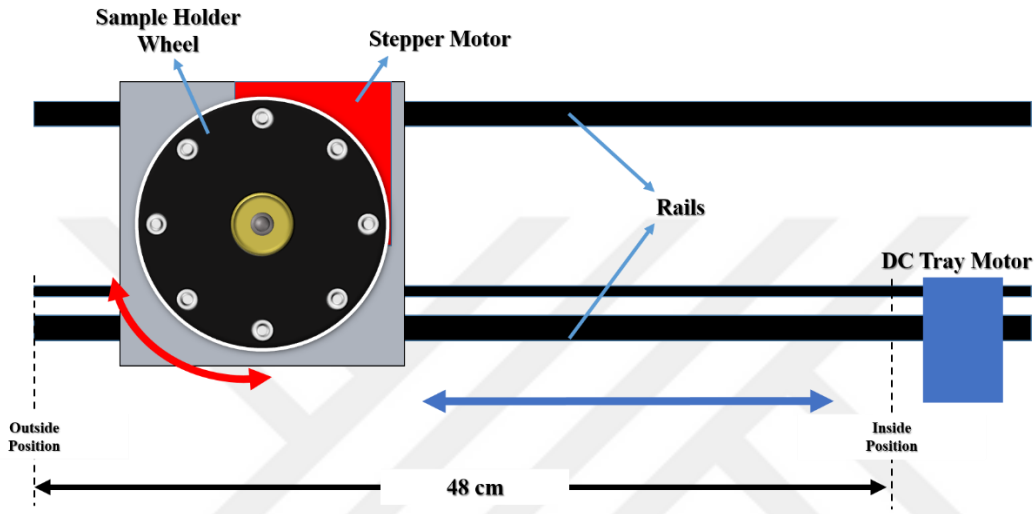



**Figure 4.3:** Simplified sketch motorized sample changer unit.

### 4.2.3 Electronic Control Unit

Both OSL measurement process (stimulation power control and photon counting) and sample positioning of the instrument are handled by an Arduino DUE single board microcontroller. Arduino DUE operates an Atmel SAM3X8E ARM Cortex-M3 CPU running at 84 Mhz. General properties of this microcontroller are mentioned in the chapter 3. In this section, hardware components and software of electronic control unit of the automated OSL reader will be explained in detail.

#### 4.2.3.1 Hardware and Connections

In figure 4.4, a block diagram of electronic control unit hardware is illustrated in order to make a better understanding of communication of components inside the OSL reader. The electronic control unit consists of Arduino DUE microcontroller, step and DC motor drivers and LED driver. Direction and speed of motors, intensity of LED is determined and sent to relevant drivers as digital logic signals by the microcontroller. The method of driving motors and LED will be explained in the next section. In addition, the microcontroller has output for user information screens such as LED indicators and 16x2 Liquid Crystal Display (LCD). The connection between the user interface software (computer) and the microcontroller is done with a serial communication (RS-232) protocol via USB cable.

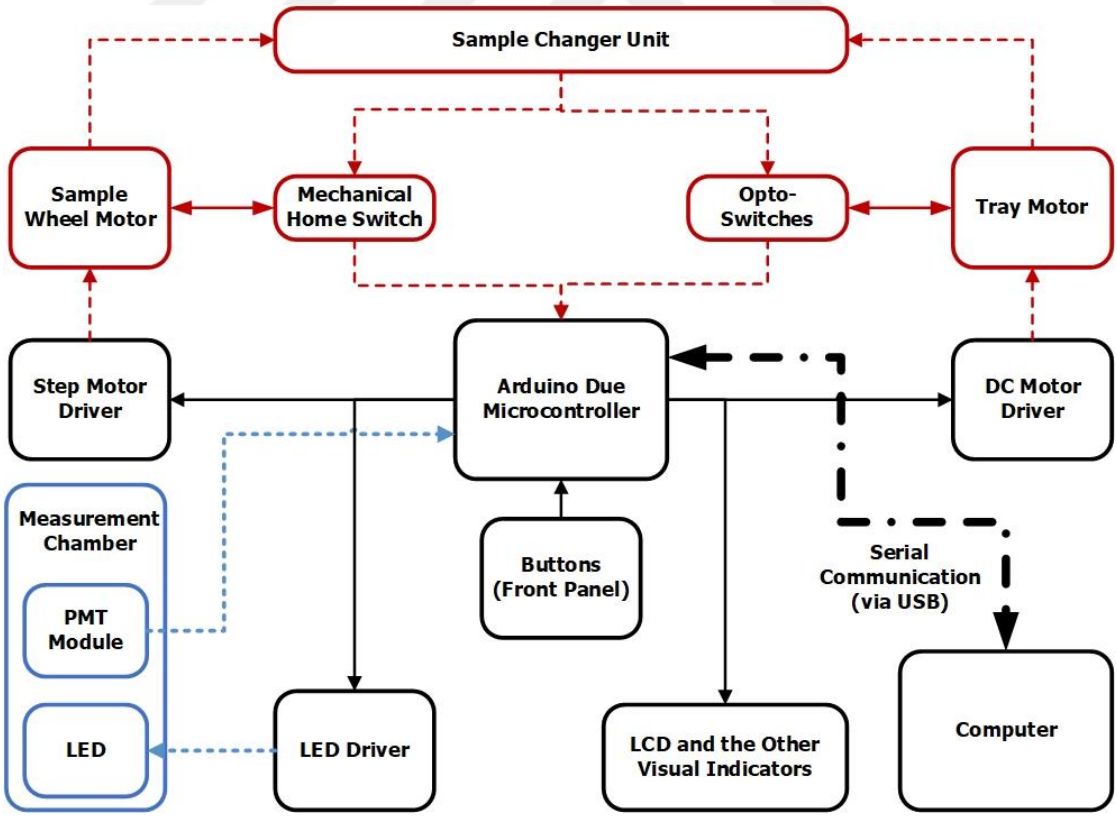



**Figure 4.4:** Block diagram of hardware of electronic control unit.

There are also inputs to the microcontroller such as switch sensors, PMT module and button inputs. A mechanical switch and two opto-switches (Honeywell, HOA0866-T55) are placed on rotary stage and tray mechanism respectively so that a reference position (home position) for the automated sample holder unit can be determined. Since the sample holder changer is controlled by the microcontroller, it constantly monitors the state of these digital output switches when sample tray is on move. Outputs of buttons, which are used for changing sample holder position and taking tray out, are also monitored by the microcontroller. The PMT module is directly connected to the microcontroller as well with a shielded cable in order to reduce noise. However, a voltage divider is placed to 5V output of the PMT module for dropping to 3.3V, since the microcontroller cannot handle input voltage greater than 3.3V (see Appendix A for Wiring Diagrams).

Furthermore, all wiring diagrams of the system is given and explained in Appendix A. The software of the electronics unit will be described in the next section.

#### 4.2.3.2 Software

A software, which is based on *C* programming language, was compiled using Arduino IDE for Arduino DUE microcontroller board. The code of the microcontroller software is given in Appendix C. The software can be analyzed in two parts. The first part handles with the operation of motorized sample changer unit and sample positioning, whereas the second part deals with OSL measurement.

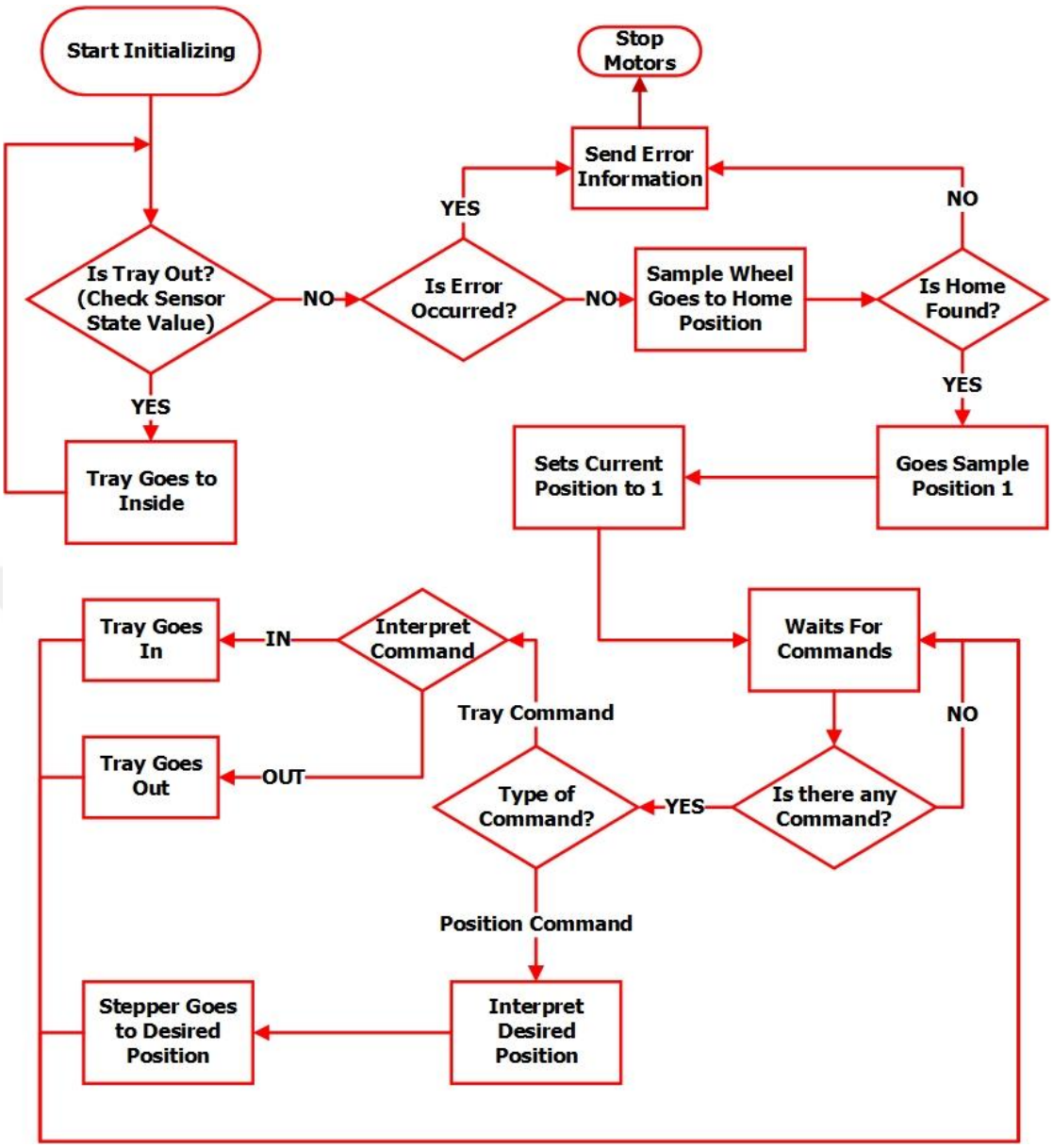



**Figure 4.5:** Flow chart of the motorized sample changer unit part of the software.

As the OSL reader is turned on, it starts with initializing mode in which sample holder position is set. In other words, the tray and sample wheel motor run to find a pre-determined home position with the help of the mechanical and optical switches (see Figure 4.4). This is done by the first part of the software. In figure 4.5, a flow chart of the motorized sample changer unit part of the software is given. When the software starts to run, it checks opto-switches in order to determine the

current position of the tray. If tray-in opto-switch is read as digital HIGH (which means the ray is not inside), the DC motor is initialized to take the tray inside until tray opto-switch is read as digital LOW. Once the tray is inside the reader, the software searches for home position of the sample wheel. This home position is determined by the mechanical switch output. The stepper motor, which is driven at 150 revolutions per minute (rpm) by the software, rotates in the clockwise direction until the mechanical switch output is read as digital HIGH (which means that the home position is found). Once the home position is found, the sample holder is moved to sample position 1 (the position where sample 1 is under measurement chamber). Then, the software goes in to idle state and waits for a command to be received for processing.

Table 4.1 gives a list of the commands that are implemented in the software for the sample changer unit. These commands can be send to Arduino using serial communication protocol.

**Table 4.1:** List of the commands implemented in the software for the sample changer unit.

| Command | Response |
|---|---|
| “TO” | Takes tray outside |
| “TI” | Takes tray inside |
| “P”+x | Goes to position x (x takes values between 1-8) |
| “RP” | Gives current tray position information |
| “PP” | Gives current sample position information |
| “HM” | Runs a initializing process (home search) |

After initializing process, the OSL reader becomes ready for measurements. The measurements process is utilized by the second part of the software. In figure 4.6, a flow chart of OSL measurement part of the software is given. The boxes with

dashed outline represents the process which is done by the first part of the software and a detailed information regarding these processes is given previously.

The OSL reader is capable of making three modes of OSL measurements as follows: CW-, LM- and TR-OSL. Before initiating the measurement, OSL mode and appropriate parameters for selected mode should be defined by the user. All parameters, which are encoded as a 37-character string, should be sent to Arduino using serial communication wire. Once, parameter values are taken by the microcontroller, all timers and counter are set for requested values. As the 'START'(ST) command is given, measurement process begins.

If a CW-OSL or a TR-OSL measurement is going to be performed, then LED intensity is set to a constant value. If LM-OSL measurement is going to be performed, then LED intensity is set to be increasing linearly from a pre-defined value to a final value which are determined as percentage of the maximum output light intensity of the LED. The LED intensity is controlled using the pulse-width modulation (PWM) technique. It is a common technique in lighting applications and advised by the LED manufacturers. Also, it is very similar approach to frequency modulation technique (Bulur et. al, 2001), which is proven to be working for controlling the average output power. In order to control LED output, an output pin of Arduino is designated for PWM signal. The frequency of the PWM signal is constant and set to be 4.2 KHz. The duty cycle of the signal can vary between 0 % to 100% (with a resolution 0.01 percent for LM-OSL stimulation light ramping) and the average power output of LED is determined by the duty cycle.

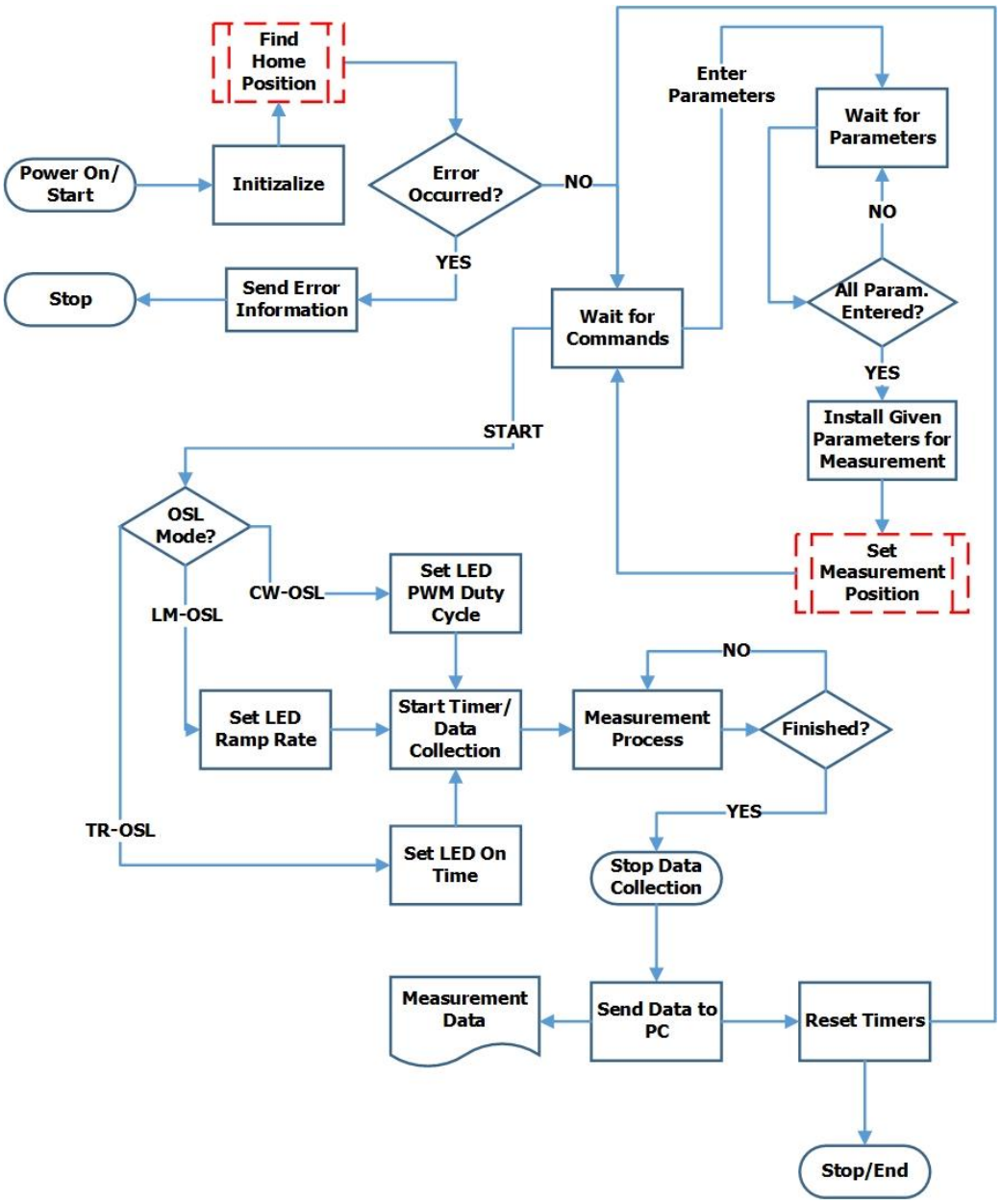



**Figure 4.6:** Flow chart of OSL measurement part of the software.

PMT module output signal works as an external clock pulses for a counter of Arduino DUE. Hence, the number of photon counts can be recorded. Also, a timer is designated for determining data collection intervals (integration time). This timer counts down from a user defined value in real time with a resolution of 1 µs. Every time the data collection interval timer value hits 0 (which means one integration time

interval has elapsed), the value of photon counter is written on the memory of the microcontroller. In this way, it is possible to have luminescence data with respect to time in the user defined time range. For example, a user wants to record the data of the number of photons emitted every 100 ms. Then, the data collection interval timer value is set to 100000 μs and the number of photons emitted every 100 ms is measured and recorded.

**Table 4.2:** List of the commands implemented in the software for the sample changer unit.

| Command | Response |
|---|---|
| "ST" | Starts the measurement. |
| "SP" | Stops the measurement. |
| "PA" | Sets the appropriate measurement parameters. A 37-character parameter string should follow. |
| "SN" | Send the data of the last measurement performed. |

As the measurement process continues, the recorded luminescence data is sent to the computer using serial communication wire. In order to prevent possible fast data transfer problems, data of measurements which has less than 10 ms integration time is transmitted to the computer after measurement is completed. Moreover, the microcontroller resets all times and counters after the completion of the measurement and make itself ready and wait for next measurement parameters. In table 4.2, a list of commands for OSL measurement part of the software is given. "ST' command stands for starting the measurement. Using this command, a previously defined (by loading measurement parameters) OSL measurement can be started. Using "SP" command, which stands for stop, all measurement process can be ceased immediately. Using "PA" command together with a following 37-character parameter string, one can upload a sequence to the microcontroller. Three examples of string for different measurement modes as follows.

Example 1: CW-OSL measurement with 80% light intensity. 100 data points to be recorded every 1 second time interval during measurement

$$\underbrace{1}_{OSL\ Type}\ \underbrace{080}_{Duty\ Cyle}\ \underbrace{xxx}_{LM-OSL\ Initial\ Duty\ Cyle}\ \underbrace{xxx}_{LM-OSL\ Final\ Duty\ Cyle}\ \underbrace{00100}_{Number\ of\ Data\ Points}\ \underbrace{0001000000}_{Integration\ Time\ (\mu s)}\ \underbrace{xxxxxxx}_{LED\ Duration\ (\mu s)}\ \underbrace{xxxxx}_{Accumulation}$$

Example 2: LM-OSL measurement with linear 0 to maximum light intensity ramping. 200 data points to be recorded every 1 second time interval during measurement.

$$\underbrace{0}_{OSL\ Type}\ \underbrace{xxx}_{Duty\ Cyle}\ \underbrace{000}_{LM-OSL\ Initial\ Duty\ Cyle}\ \underbrace{100}_{LM-OSL\ Final\ Duty\ Cyle}\ \underbrace{00200}_{Number\ of\ Data\ Points}\ \underbrace{0001000000}_{Integration\ Time\ (\mu s)}\ \underbrace{xxxxxxx}_{LED\ Duration}\ \underbrace{xxxxx}_{Accumulation}$$

Example 2: TR-OSL measurement with 50 ms LED ON duration. 500 data points to be recorded every 1 ms time interval during measurement. The measurement will be repeated 1000 times.

$$\underbrace{2}_{OSL\ Type}\ \underbrace{100}_{Duty\ Cyle}\ \underbrace{xxx}_{LM-OSL\ Initial\ Duty\ Cyle}\ \underbrace{xxx}_{LM-OSL\ Final\ Duty\ Cyle}\ \underbrace{00500}_{Number\ of\ Data\ Points}\ \underbrace{0000001000}_{Integration\ Time\ (\mu s)}\ \underbrace{050000}_{LED\ Duration}\ \underbrace{01000}_{Accumulation}$$

Finally, “SN” command can be used for requesting the measurement data after measurement.

### 4.2.4 User Interface (PC Software)

It is crucial that instrumentation systems have a user friendly and clearly designed control and data collection software. In that sense, a user interface software is written for the OSL reader. *Python* programming language (Rossum, 1995) is used for design. As mentioned before, there is a great number of libraries available for *Python*, since it is frequently used in high-level software programming. *TkInter* (https://wiki.python.org/moin/TkInter, Accessed on April 11th, 2017), *matplot* (Hunter, 2007), *NumPy* (http://www.numpy.org/, Accessed on April 11th, 2017) and *Python Imaging Library* (http://www.pythonware.com/products/pil/, Accessed on April 11th, 2017) libraries are utilized for coding this user interface software.

In figure 4.7, a screen shot of user interface software is given. This software enables users to create measurement sequences for consecutive OSL measurements and run them using the 'Setup' window. It is also possible to observe ongoing luminescence measurement (luminescence intensity against measurement time plot) in near-real time under the 'Live Plot' window.

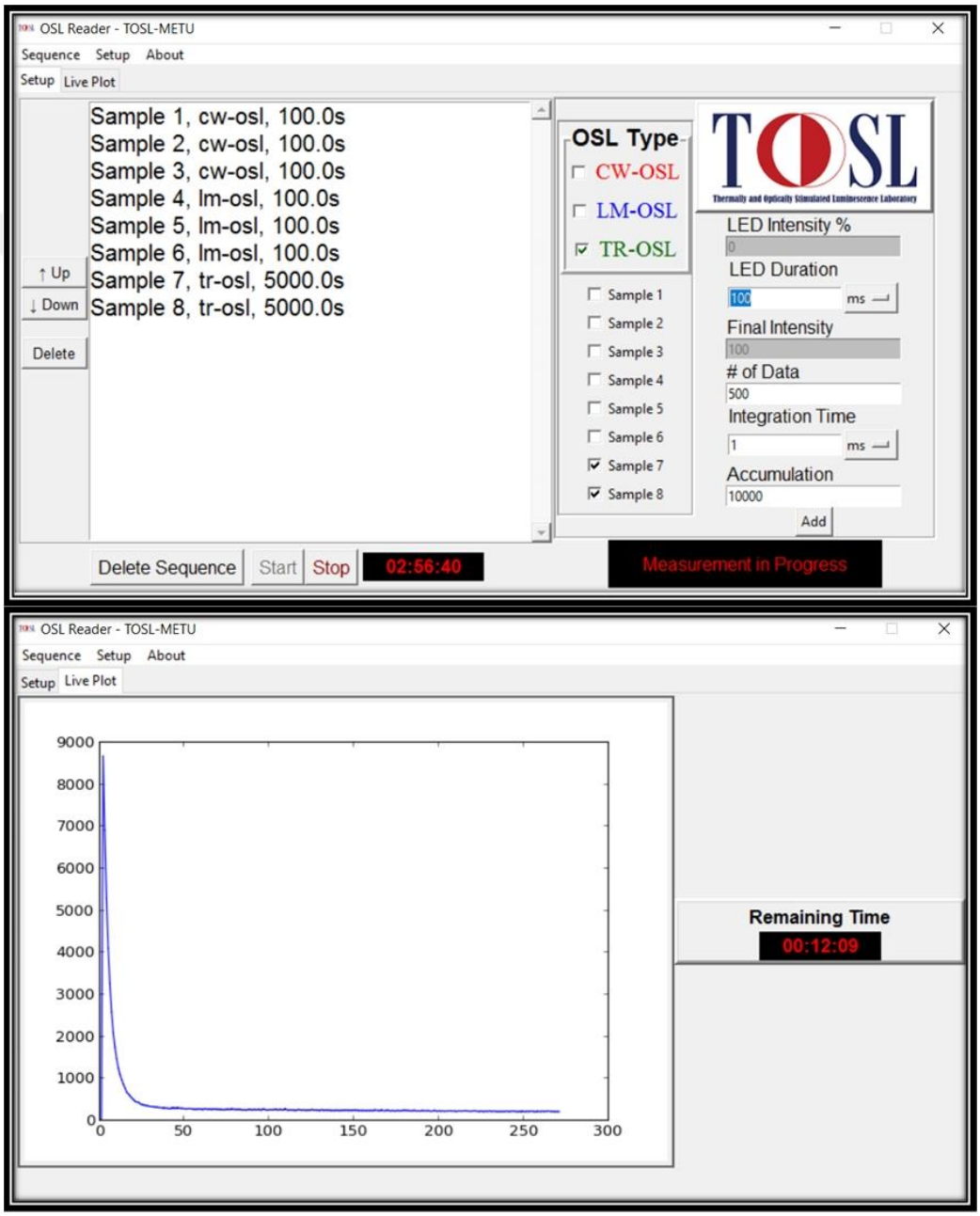



**Figure 4.7:** Screenshot of user interface software.

The user interface software keeps track of ongoing measurement, send appropriate commands to the microcontroller, obtains the measurement data from the microcontroller and saves the data file for further analysis. A flow chart of user interface software is given in figure 4.8. Once the software is executed, a communication port should be selected by the user so that serial communication between the OSL reader and the computer can be initialized. It can be selected using 'Setup' drop down menu. After the selection of communication port, a sequence should be created or loaded. In order to load a previously saved measurement sequence, 'Sequence' drop down menu should be used. Measurement sequences can be created using the 'Add' button. User should select desired the OSL mode, samples to be measured and fill relevant entry boxes for that measurement mode in order to add a step to the sequence. These steps can be observed in the list box. It is possible to delete a step or change the order of measurement using 'Delete', 'Up' and 'Down' buttons located in the setup window.

Once a measurement sequence is created or loaded, 'Start' button becomes active. As measurement process is started, a pop-up dialog box is opened and asks user to write a file name and location for measurement data. As they are entered, the software sends position command to the microcontroller and waits until the microcontroller gives the information that desired sample position is ready. Then, all measurement parameters are encoded to 37-character string and sent to Arduino. Luminescence data, which is retrieved from the microcontroller during (or 'after', depending on data collection interval – explained in the microcontroller section) measurement, is recorded to a data file. The data file, which is saved to desired location at the end of the measurement, also includes measurement parameters and sample number in order to avoid possible confusion during a future data analysis.

The code of the user interface software is given in Appendix D.

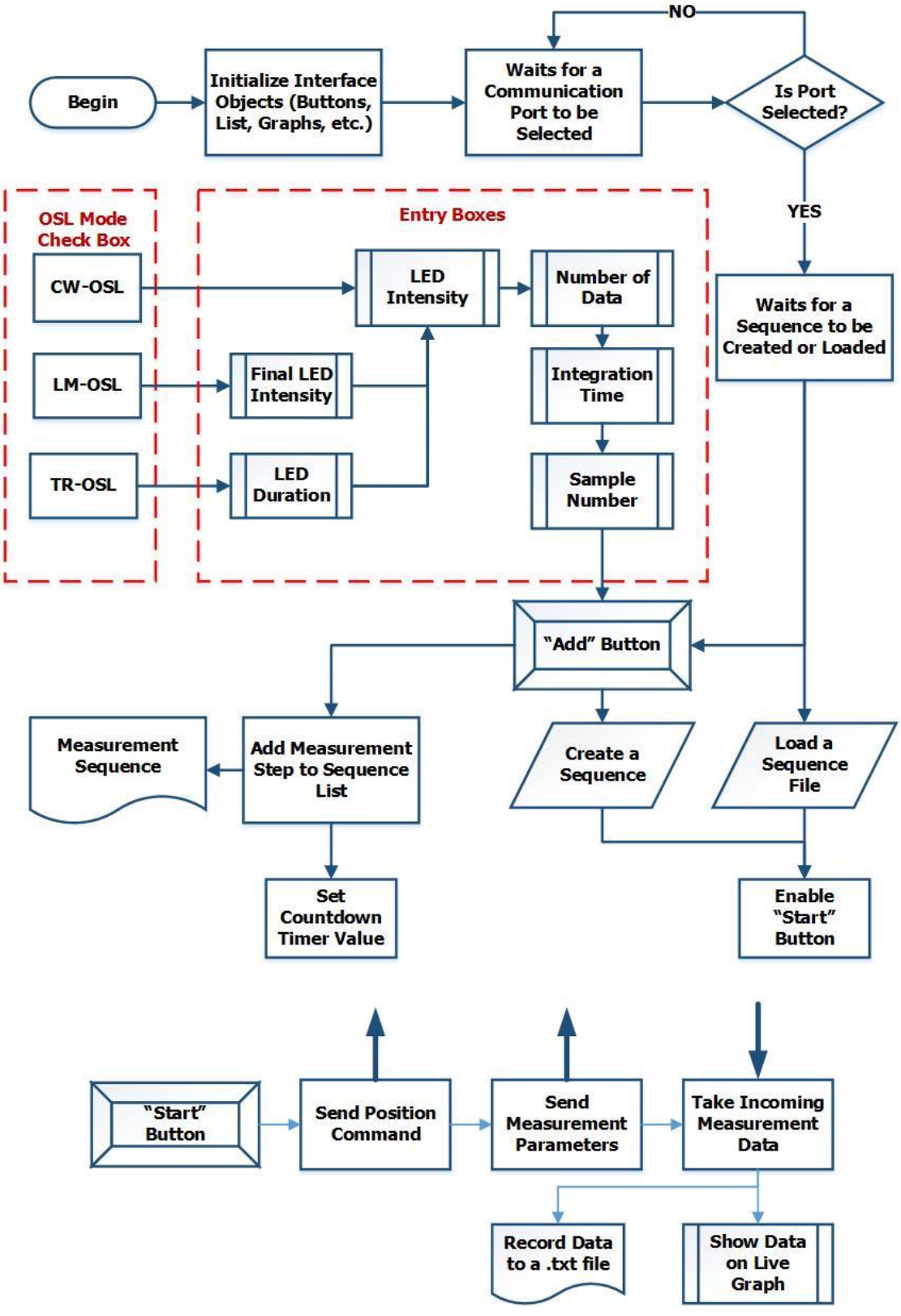



**Figure 4.8:** Flow chart of User Interface Software

## 4.3 Principles of Measurement Modes

### 4.3.1 CW-OSL

A CW-OSL measurement can be added to the measurement sequence. In order to add CW-OSL measurement step, users should check samples to be measured and enter LED intensity, number of data and integration time values. LED intensity is defined as percentage of maximum output power of LED and can have a value between 1-100. Number of data value specifies how many data points will be collected. It can have a values between 1 and 65535. The separation between the data points is defined by integration time value. Integration time can have a value between 1 μs and 10 s.

### 4.3.2 LM-OSL

A LM-OSL measurement can be added to the measurement sequence. In order to add CW-OSL measurement step, users should check samples to be measured and enter LED intensity, number of data and integration time values. In this mode, LED intensity values defines the initial stimulation intensity of LM-OSL. In addition, there is a final intensity entry which should be filled by the user. The linear increase of the stimulation light during the measurement is between these initial and final values of intensity.

### 4.3.3 TR-OSL

A TR-OSL measurement can be added to the measurement sequence. In order to add CW-OSL measurement step, users should check samples to be measured and enter LED intensity, number of data and integration time values. In addition, there is a LED duration value which should be entered by the user. This value defines the length of the stimulation and can have a values between 100 μs to 1 s.

# CHAPTER 5

# RESULT AND DISCUSSION

In this chapter, experiments done for determination of optical and/or electronic characteristics of measurement system is explained in detail. In addition, results of OSL measurements of some materials relevant for dosimetry in different measurement modes are also given.

## 5.1 Determination of the Stimulation LED Characteristics

As it was mentioned in Chapter 2, characteristics of stimulation light is a very significant parameter for OSL measurements since energy required for emptying traps is provided by it and the de-trapping rate of electrons are considered to be proportional to the light intensity. Hence, stable output of stimulation light (both in terms of wavelength and power) is a must in order to have an accurate OSL measurement system. Time response of LED is also important for TR-OSL. For this reason, some measurements were performed in order to determine the characteristics of stimulation LED (Cree XQEBLU-SB-0000-000000Y01). For the measurements presented in this section, a calibrated power meter (Newport Model 841), fiber-coupled spectra-radiometer (International Light, RPS900-R), a signal generator (OWON AG2052F), a digital oscilloscope (GW Instek GDS-2204), a DC voltage source (Trio Model PR-554), a voltmeter (TT T-ECHNI-C MV-64 Multi-meter) and an ammeter (Keithley 199 DMM) was used.

It is known that the emission spectrum of an LED depends on the current passing through (Bøtter-Jensen et al., 2000). When emission spectrum measurements of LED are conducted, it was observed that there is wavelength shift. In figure 5.1, emission spectra of the stimulation LED are given for 10 mA (dashed-line), 160 mA (dotted-line) and 340 mA (solid line). Peak wavelength of the emitter for 10 mA is 467 nm, whereas 340 mA is 464 nm. The full width half maximum of emission is 21 nm. Even though, the shift is only 4 nm, it was taken in consideration due to the wavelength dependency of OSL in the design of the measurement electronics so that the current flow through the LED is constant.

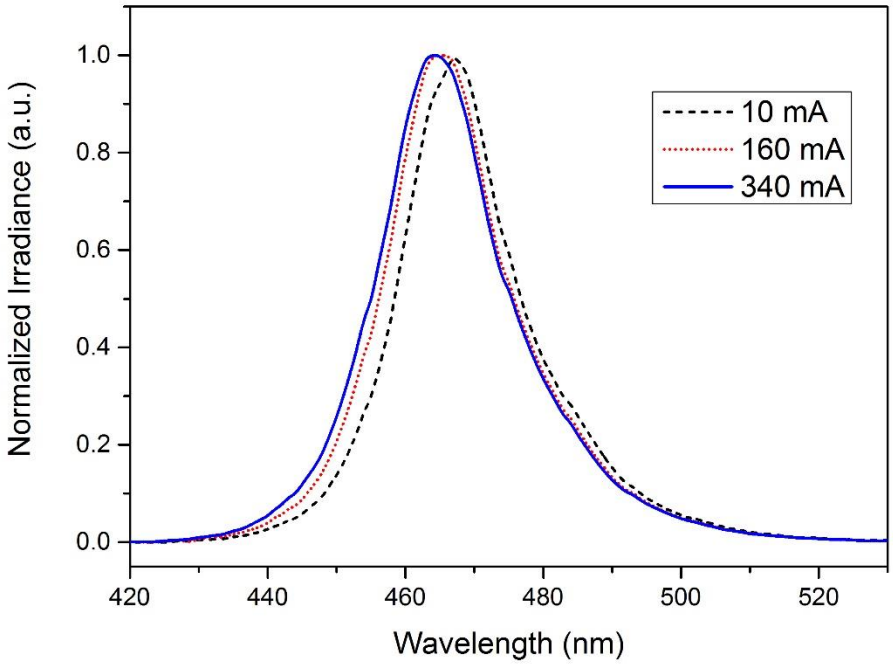



**Figure 5.1:** Emission spectra of the blue stimulation LED at different current values.

Dependence of the output power of LED on the current passing through was also measured. As it can be seen from figure 5.2, the relation between output power and current is almost linear. Maximum power achievable when driving LED in the recommended range of current is 134 mW. The detector of the power meter is placed

13 cm away from the LED for this measurement. This is the distance between LED and sample holder of OSL measurement system. This measurement also confirms the importance of the driving LED at a constant current value and changing intensity of LED using other than changing current passing through.

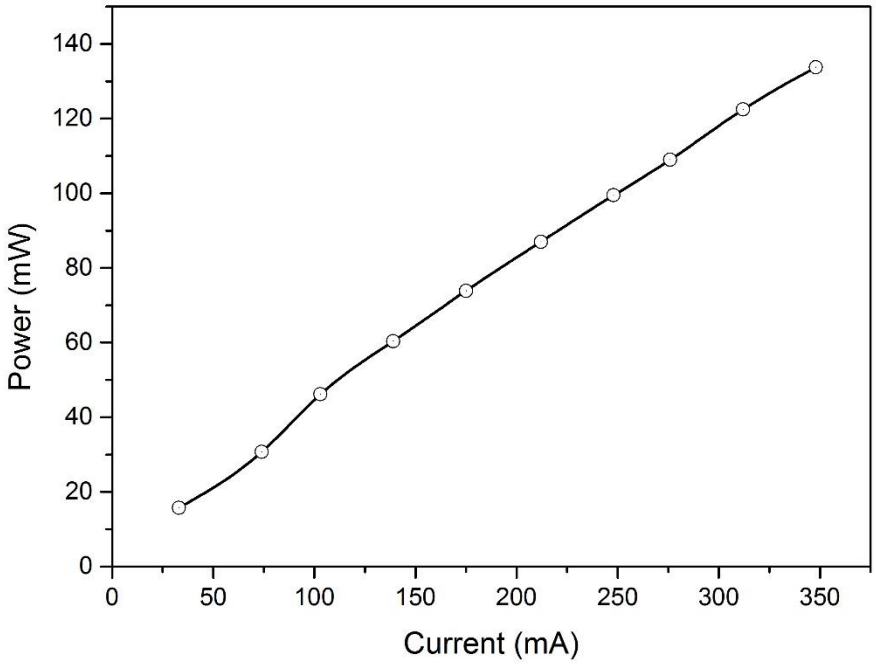



**Figure 5.2:** Output power of the blue LED with respect to current passing through.

Another important parameter for stability of the LED output is temperature of the emitter. The efficiency of the LED is changing with the temperature. Therefore, a thermocouple was placed under the printed circuit board (PCB) of the LED and temperature of the PCB board was measured with respect to current passing through the emitter. As it can be seen form figure 5.3, the PCB temperature is constant from 70 to 350 mA. The temperature recording presented was taken using TT T-ECHNI-C MV-64 multi-meter after 10 minutes of operating LED.

Since the LED is driven with a constant voltage supply rather that constant current supply, the relationship between current passing through the LED and the

voltage between the sides of the junction is another parameter that is significant. Inset to figure 5.3, potential difference values with respect to current is given. It was determined that in order to get maximum output power from the LED, 3.00 V should be applied.

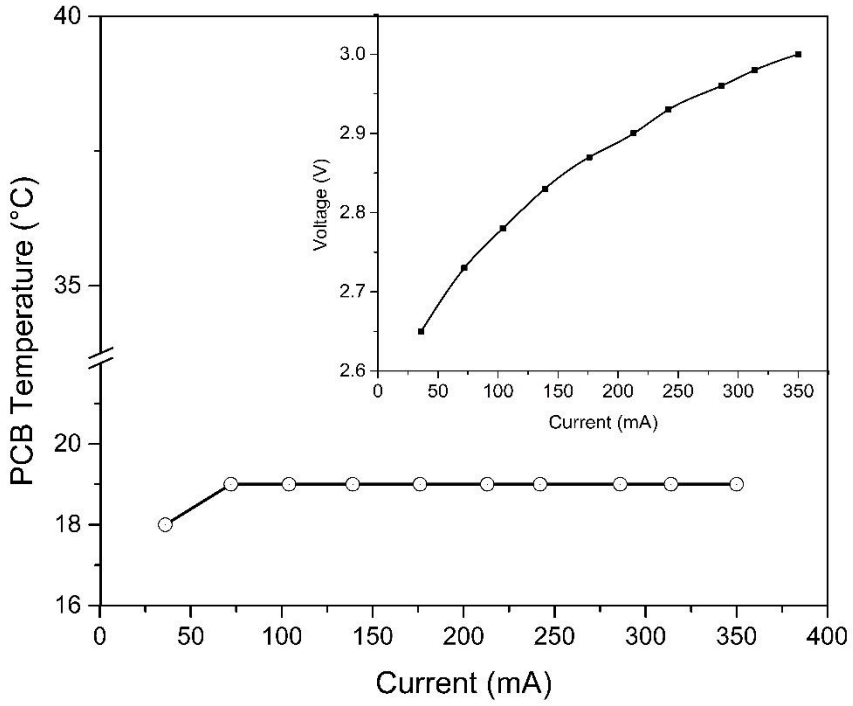



**Figure 5.3:** The PCB temperature of the blue LED after 10 minutes of operation time with respect to current passing through. Inset: The relationship between voltage applied to the junction and current passing through.

The time response determines how small pulses can be generated using the LED and it is significant for TR-OSL measurements. For this reason, rise and fall time of the LED were measured. A photodiode (Hamamatsu S1336-8BQ, Rise Time = 100 ns) was placed in front of the LED (driven with pulses) and the output signal of the photodiode (together with paralelly connected 100 Ω terminator resistor) was observed using an oscilloscope. In figure 5.4 the output signal of the photodiode is given with respect to time. The rise and fall time of the LED (together with its driver)

was found to be 1.02 µs and 1.22 µs respectively. In the calculation, rise time and fall time was considered to be the time elapses between 10% and 90% of the output signal.

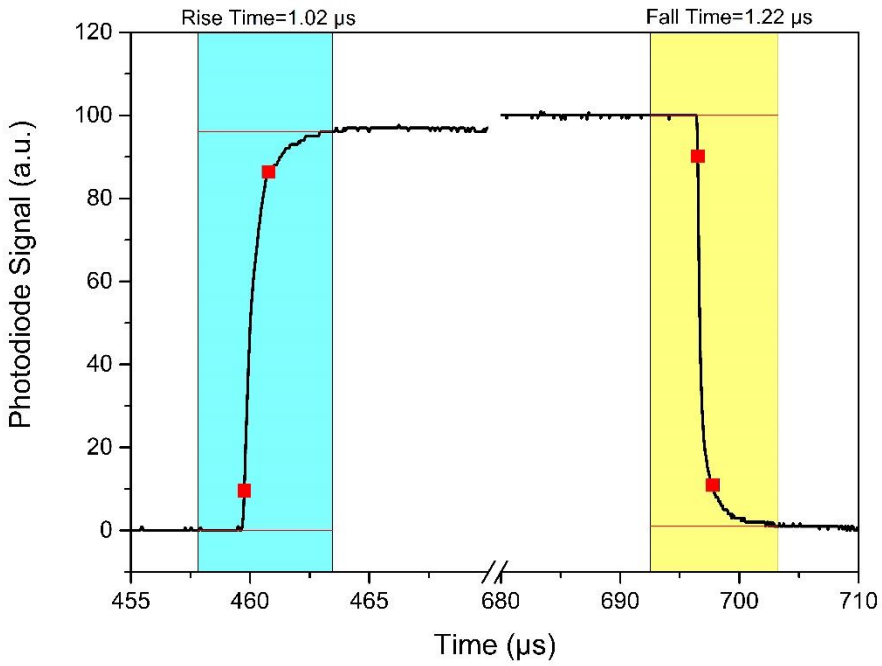



**Figure 5.4:** Output Signal of Photodiode in Time for a LED Pulse.

As a result of these measurements, it was determined that the current passing through the LED should be constant for the stability of the output. For this reason, the LED driver circuit was designed in a way that constant 350 mA current is passing through LED (see Appendix A). Moreover, it is determined that stimulation pulses in µs regime can be generated when the device is used in TR-OSL mode.

As mentioned in the previous chapter, the stimulation light intensity is controlled using PWM technique. In order to verify the linear relationship between duty cycle of the signal and intensity output, illumination power drops on the sample is measured after the completion of instrument. As it can be seen from figure 5.5, the relationship is linear and the output power of LED on sample varies from 0.8 mW to 91.4 mW as duty cycle changes from 1% to 100%.

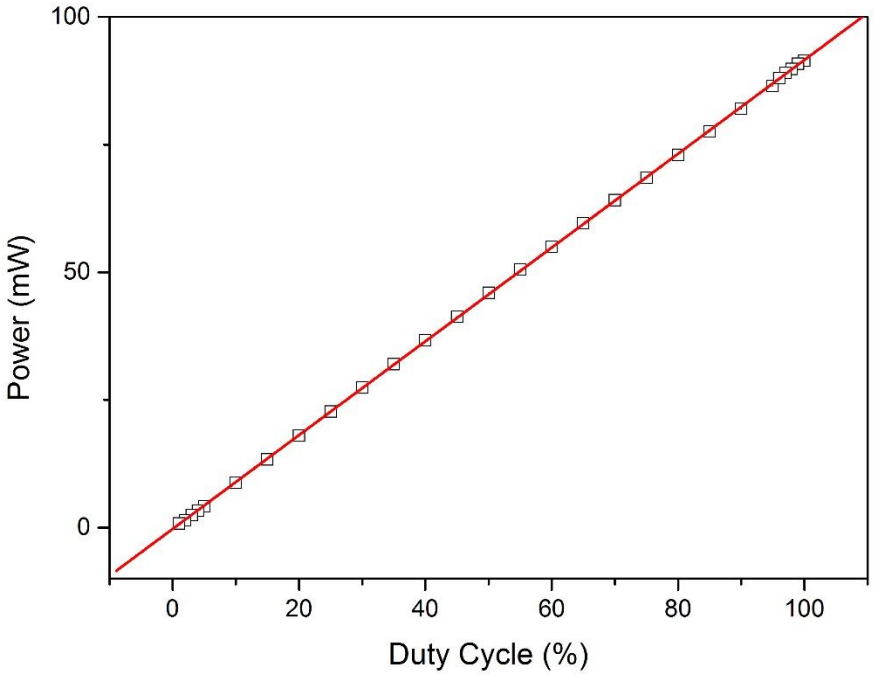



**Figure 5.5:** Output power of the LED on sample with respect to duty cycle.

## 5.2 Determination of Measurement Chamber Characteristics

As mentioned in Chapter 2, it is very significant in OSL measurements that stimulation and emission light should be separated well. For this reason, the OSL measurement system is designed in a way that emission of luminescence in the near-UV region of the electromagnetic spectrum where the stimulation light is blue. The transmission and reflection spectra of the filters (including the dichroic mirror) was measured and presented in this section. In order to make these measurements an experimental setup wass prepared. A simplified sketch of the setup is given in figure 5.6.

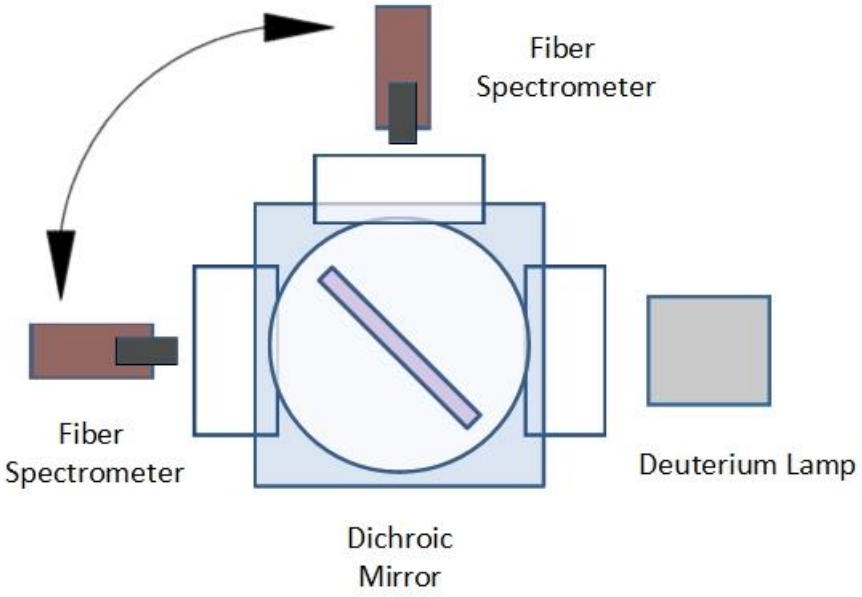



**Figure 5.6:** A simplified sketch of the experimental setup used for reflection and transmission measurements.

In this setup, measurement chamber of OSL reader was placed between the detector of fiber spectrometer (International Light, RPS900-R) and a deuterium lamp source (Hitachi 890-2430). The reason for the use of $D_2$ lamp is to measure the reflectance spectra of dichroic mirror in the near-UV region. Firstly, spectrum of light source was recorded as *input*. Then, reflection/transmission spectrum was recorded as *output* after placing dichroic mirror. Percentage reflection and transmission spectra were obtained using the formula $(\frac{output}{input} x100)$.

In figure 5.7, transmission spectrum of the dichroic mirror is given together with emission spectrum of the stimulation LED. The intensity of LED emission is normalized between 0 and 1. The transmission is around 70% in the visible region and below 0.5% at 400 nm. This is preferred since the stimulation of samples done with blue LED which peaks in the visible region.

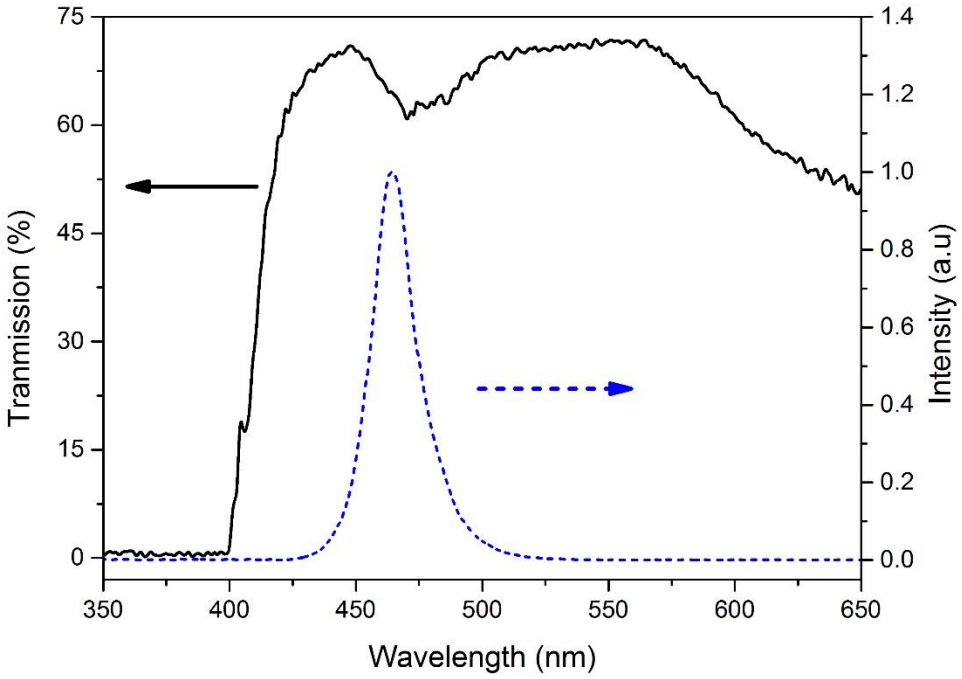



**Figure 5.7:** Transmission spectrum of dichroic mirror (solid-line) and emission spectrum of LED (dashed-line). [Transmission spectrum of the dichroic mirror is measured at 45° incident angle]

Moreover, transmission spectrum of UV filter pack, reflection spectrum of dichroic mirror and combination of these spectra were measured using the same setup given in figure 5.6. As it can be seen from the figure 5.8, reflection of the bare dichroic mirror is around 36% at 365 nm. Combination of mirror and filter pack results for the system to have 18% overall efficiency at the maximum. Total efficiency of the measurement detection can only be calculated after the consideration of efficiency of PMT module. Quantum efficiency of PMT at 375 nm is around 16% according to the datasheet. Hence total efficiency of detection can be roughly estimated as 3%.

As a result of these measurements, the stimulation and detection regions of the system is verified to be separated from each other.

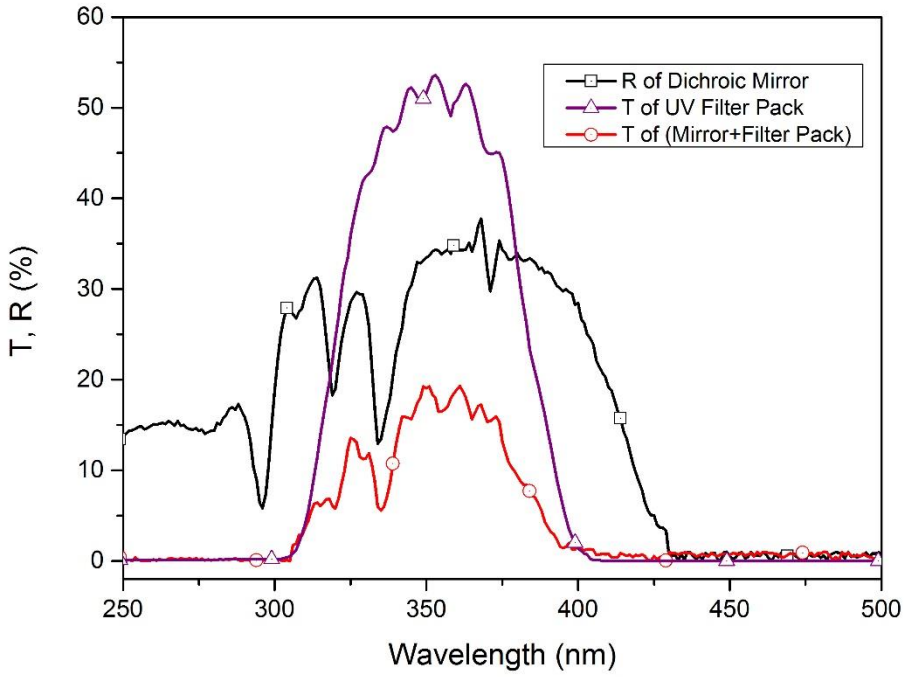



**Figure 5.8:** Reflection spectrum of dichroic mirror (black/square). Transmission spectrum of the UV filter pack (purple/triangle). Transmission spectrum of dichroic mirror and UV filter pack combination (red/circle). [Reflection and transmission spectra of the dichroic mirror are measured at 45° incident angle]

After the completion of the OSL reader, background measurements were conducted in order to determine the stability of the measurement system. For this purpose, background was measured and recorded in one second intervals for 200 minutes when stimulation light source was on (at maximum power) and off (see figure 5.9). For both measurements, background data points (represents counts per second) were averaged. It was found that averaged background of light-on measurement was 44.48 counts per second with standard deviation of 2.30 and it was 1.45 counts per second with standard deviation 0.66 when light was off.

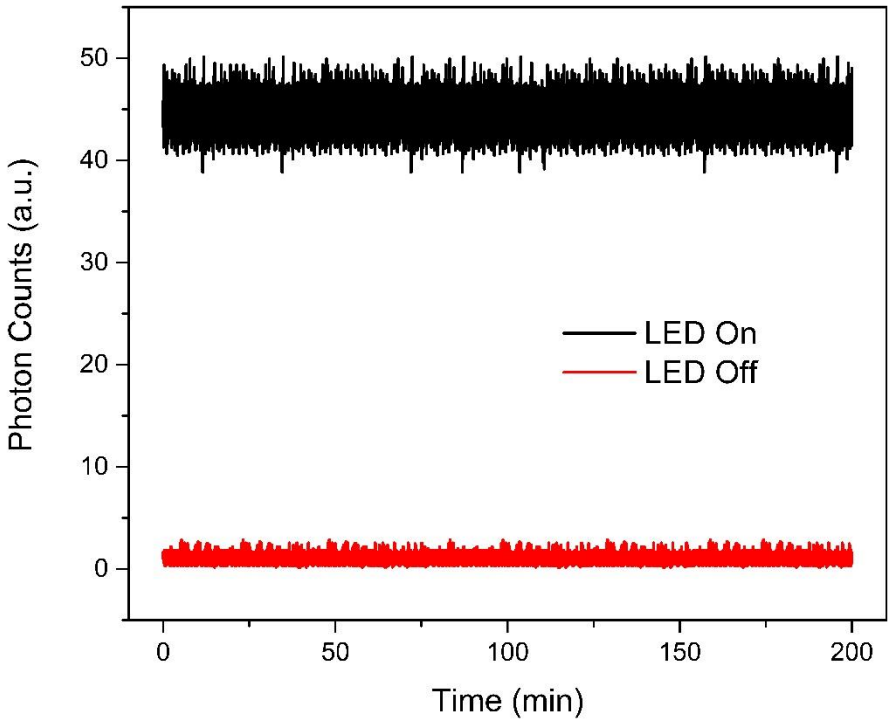



**Figure 5.9:** Background measurements of OSL measurement system when stimulation light is on and off.

### 5.3 Test Experiments

Functionality and performance of the OSL measurement system were tested using materials relevant to dosimetry and dating. Materials used for these tests include α-$Al_2O_3$:C crystals (5 mm diameter and 1 mm thickness, Landauer Inc.), Beryllia (BeO) ceramics (4mm diameter and 0.8 mm thickness, Thermalox 995, BrushWellman Inc.), heated natural quartz ($SiO_2$) grains (250-355 µm, Merck) and gem quality zircon (100 µm, $ZrSiO_4$). A picture of these samples after preparation is given in figure 5.10. Samples were irradiated using $^{90}Sr/^{90}Y$ beta source (Amersham International) with a dose rate around 25 mGy/s.

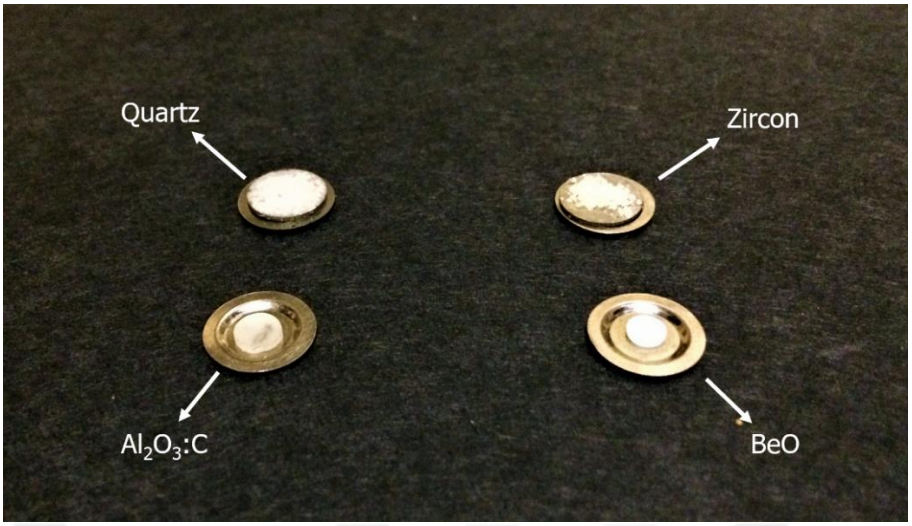



**Figure 5.10:** A picture of samples used for OSL measurements.

Samples were prepared for measurement using appropriate annealing procedures given in table 5.1. Due to heating, color of the zircon samples (originally brown) was removed and they were observed to have light orange color. After heating, these zircon samples were crushed and ground in an agate mortar to a size around 100 µm. During OSL measurements $Al_2O_3$:C and BeO samples used as they are. However, a single layer of quartz grains and a single layer of zircon grains were prepared and attached to aluminum disks of diameter 10 mm using silicone oil.

**Table 5.1:** Annealing temperature and durations of samples before OSL measurements.

| **Material** | **Annealing Temperature (°C)** | **Annealing Duration (minutes)** |
|---|---|---|
| $Al_2O_3$:C | 900 | 15 |
| BeO | 650 | 30 |
| Quartz | 500 | 30 |
| Zircon | 900 | 15 |

In order to eliminate the luminescence contribution originating from shallow traps, all samples were pre-heated at 100°C for 15 min and cooled back down to room temperature between irradiation and measurement. For pre-heating, the samples were placed inside an aluminum case which is kept inside a hot sand oven whose temperature was controlled within ±1°C.

Test experiments were conducted using three measurements modalities namely CW-, LM- and TR-OSL. In the following sections, a brief summary of the results of these experiments are given. OSL curves obtained with different measurement modalities were analyzed using deconvolution by curve-fitting using appropriate model functions given in Table 5.2. The non-linear curve fitting is based on the minimization of the $\chi^2$ function using Levenberg-Marquardt algorithm (Levenberg, 1944; Marquardt, 1963). For this purpose, a commercial software package was used (OriginLab, Massachusetts, USA).

**Table 5.2:** Model functions used in curve-fitting for different OSL modalities.

| OSL Mode | Function |
|---|---|
| CW-OSL | $I_{CW-OSL}(t) = \sum_{i=1}^{k} I_{0i} e^{-b_i t} + B$ |
| LM-OSL | $I_{LM-OSL}(t) = \sum_{i=1}^{k} A_i b_i \frac{t}{P} \exp\left(-\frac{b_i t^2}{2P}\right)$ |
| TR-OSL | $I_{TR-OSL}(t) = \sum_{i=0}^{k} I_{oi} e^{-\frac{t}{\tau_i}} + B$ |

### 5.3.1 CW-OSL Mode

CW-OSL measurements were performed using the materials mentioned above in order to verify the OSL detection of system in this mode. In figure 5.11, a decay curve of 100 mGy irradiated $Al_2O_3$:C (which is widely used commertial dosimeter) disk is given with the background measurement. The background measurement, which is the second OSL measurement of the same sample, was done in order to verify that the measured decay curves are genuine OSL signal due to depletion of trapped charges.

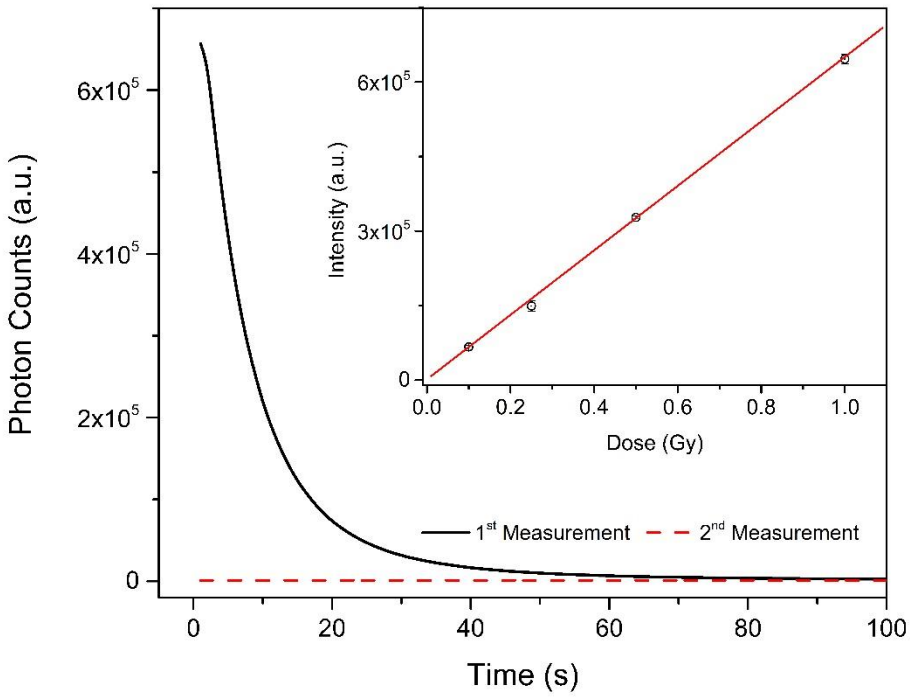



**Figure 5.11** Decay curves of $Al_2O_3$:C (100 mGy irradiated) taken from consecutive measurements [(line) is first measurement, (dashed) is second measurement]. **Inset:** Dose response of $Al_2O_3$:C in the range from 0.1 Gy to 1 Gy.

Inset to figure 5.11, dose response of $Al_2O_3$:C in the dose range from 0.1 Gy to 1 Gy is given. Signal intensities of different doses were obtained by deconvolution using curve-fitting to model function. For every dose in dose response graph, three measurements are done and their results are averaged. Error bars shown in the plot represent the standard deviation of this averaged value. The dose response of the sample was observed as linearly increasing in that range. It has been previously reported that this material shows linearity in the dose response up to 2 Gy (Yukihara, et al. 2014). Hence, the results obtained were consistent with the literature and it can be stated that CW-OSL signals from $Al_2O_3$:C can be detected using the constructed OSL measurement system as expected.

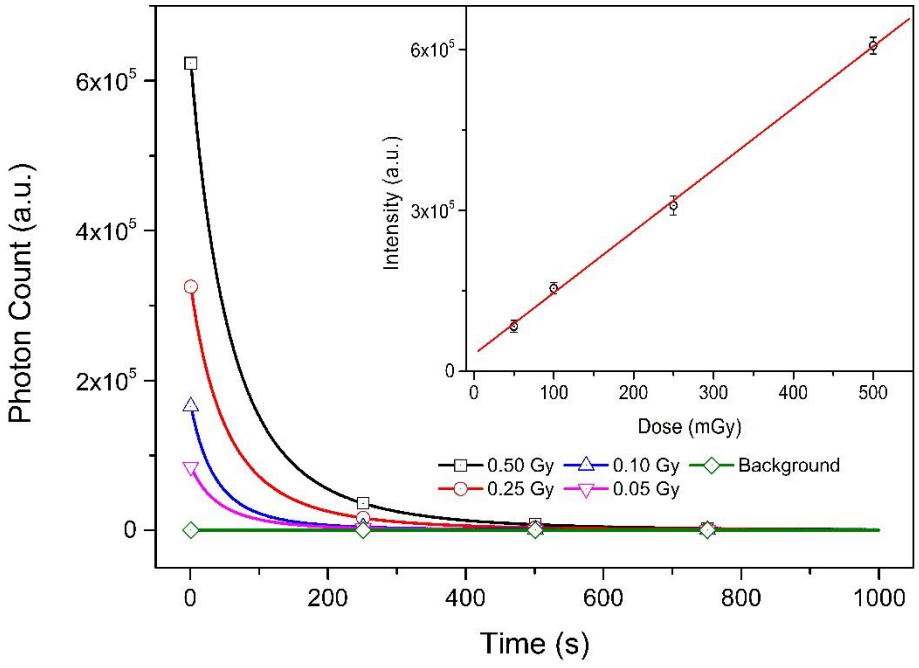



**Figure 5.12:** Decay curves of BeO chips with different doses: 0.50 Gy (black/square), 0.25 Gy (red/circle), 0.10 Gy (blue/triangle), 0.05 Gy (magenta/reverse triangle) and background (green/diamond). Inset: Dose response of BeO chips in the interval 50 mGy to 500 mGy.

BeO, which is one of two commercial dosimeters available today, was used for CW-OSL test measurements. In figure 5.12, decay curves of BeO chips

irradiated with different doses are given together with background signal. Background signal is a repeated measurement ($2^{nd}$ measurement) on 500 mGy irradiated sample. Dose response of BeO chips can also be seen inset to figure 5.12. Signal intensities of different doses were obtained by deconvolution using curve-fitting. For every dose in dose response graph, three measurements are done and their results are averaged. Error bars shown in the plot represent the standard deviation of this averaged value. It was reported by several authors that dose response of BeO chips show linearity from 5 μGy to 5 Gy (see Yukihara and McKeever, 2011 for a review). The measurements done with the OSL reader shows this dose response linearity as well in the given range (50 – 500 mGy).

The instrument's minimum detectable dose (MDD) together with dosimeter is another parameter to be calculated when one is investigating OSL measurements of personal dosimetry materials like $Al_2O_3$:C and BeO. One can calculate the MDD using 3σ method. In order to detect a dose using CW-OSL measurement, the signal intensity should be greater than the sum of background and three times its standard deviation (3σ) (Yukihara and Mckeever, 2011). Using this approach, it is possible to say that MDD using the constructed OSL device is less than 100 μGy for $Al_2O_3$:C and BeO. Even it can be enhanced 2.5 times for BeO and 4 times for $Al_2O_3$:C by increasing stimulation light intensity.

Another material used for CW-OSL measurements was quartz which is one of the most preferred materials for luminescence dating and retrospective dosimetry. As an example of CW-OSL decay curve of irradiated (100 Gy) quartz sample is given in figure 5.13. As seen from figure, the OSL from quartz exhibit a fast decay with a slower tail extending to around 250 seconds. Moreover, most of the signal is erased after $1^{st}$ measurement and $2^{nd}$ measurement is very close to the background signal. Inset to figure 5.13, dose response of quartz is also given in the range of 50 Gy to 250 Gy. The dose measurements were repeated three times. The error bars represented in the plot are standard deviation of averaged intensities. In the dose range used, OSL signal intensity was observed to be increasing linearly. The saturation of luminescence intensity was reported for high dose irradiated quartz

(Lowick et. al, 2010). In order to observe this saturation, it is clear that dose response measurements of the material should be extended to higher doses.

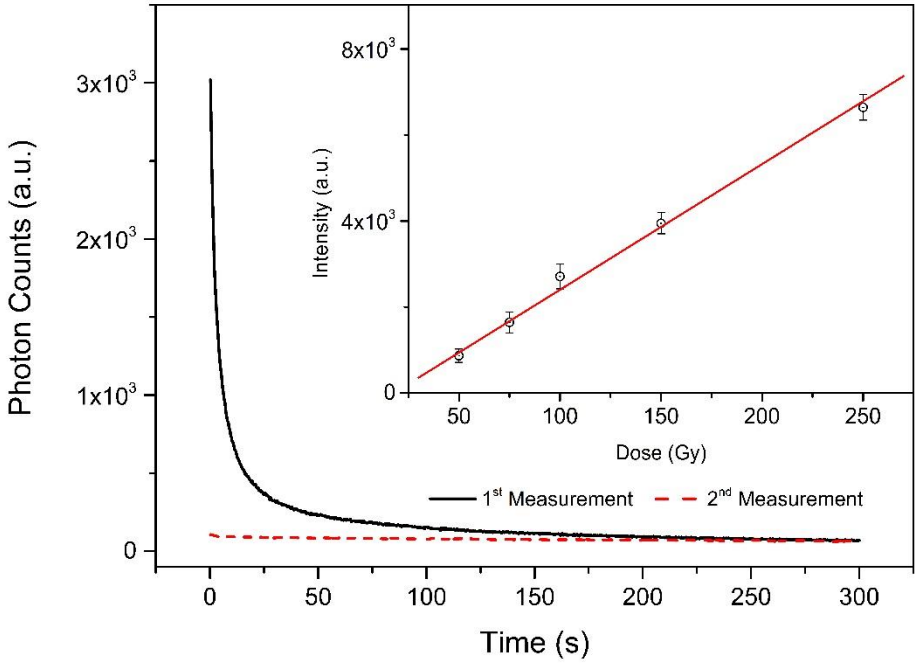



**Figure 5.13:** CW-OSL decay curves of quartz (100 Gy irradiated) taken from consecutive measurements [(line) is first measurement, (dash) is second measurement]. **Inset:** Dose response of quartz from 50 Gy to 250 Gy.

Gem quality brown zircon ($ZrSiO_4$) crystals collected from Cambodia were also measured using CW-OSL mode. The preparation process of the samples before measurements is described in the previous section. An example of the OSL decay curve from an irradiated (25 Gy) zircon sample is shown in figure 5.14. OSL curve of this samples were analyzed since there was no CW-OSL study reported for this material. As it can be seen from the figure, zircon exhibits a bright, fast decaying luminescence signal. The OSL decay curve can be approximated using a linear combination of three exponential decay functions with decay constants of 46.32 s, 7.22 s and 0.79 s (see inset of figure 5.14).

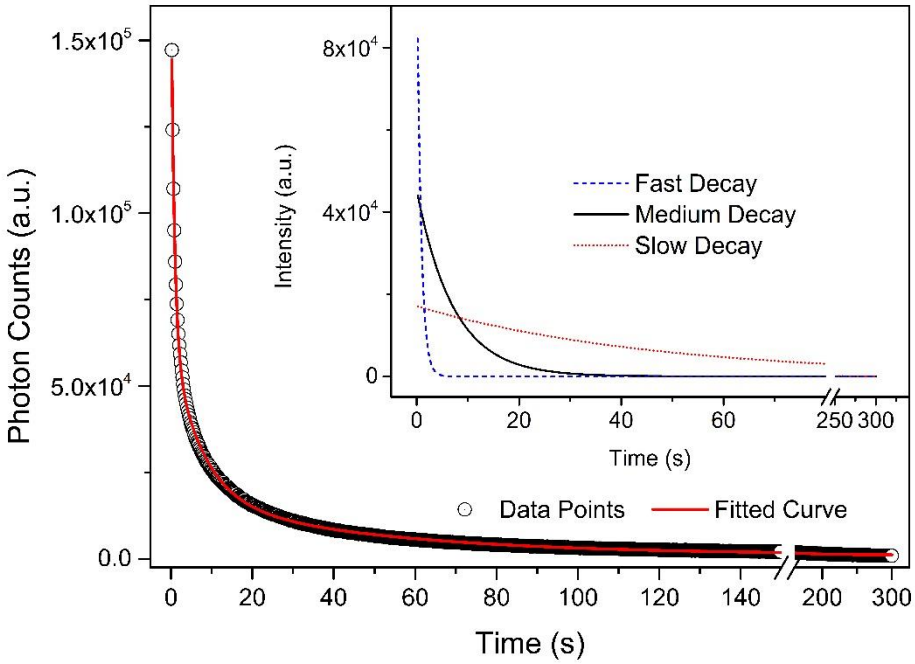



**Figure 5.14:** CW-OSL signal from zircon samples fitted with a linear combination of three exponential decay curves. Inset: Components of the OSL decay curve: Fast Decay (dashed/blue line), Medium Decay (solid/black line) and Slow Decay (dotted/red line).

Figure 5.15 shows OSL decay curves of zircon samples irradiated at various doses (5, 10, 25 and 50 Gy) together with the background signal which was obtained by a repeated measurement of the 25 Gy irradiated sample. Dose response of zircon samples can also be seen inset to figure 5.15. Signal intensities of different doses were obtained by deconvolution using curve-fitting. Intensity represents the sum of three components of the decay. OSL measurements were repeated three times and error bars shown in the plot represent the standard deviation of averaged intensities. The CW-OSL measurements performed shows that dose response was linear in the range between 1 Gy to 50 Gy. Linear increase of luminescence with the dose was shown by Bulur et. al. (2014) for the same samples. Even though TR-OSL technique

was employed on that study, it may be said that CW-OSL measurements show consistency with the study on the samples in terms of dose response.

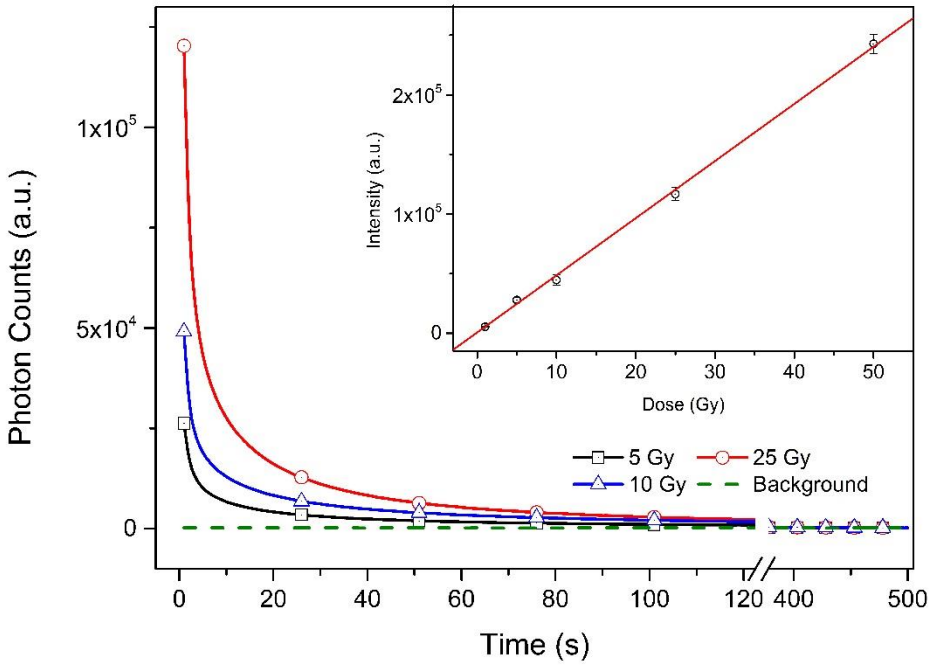



**Figure 5.15:** Decay curves of zircon samples with different doses: 5 Gy (black/square), 10 Gy (blue/triangle), 25 Gy (red/circle) and background (green/dashed). Inset: Dose response of zircon samples chips in the interval 1 Gy to 50 Gy.

### 5.3.2 LM-OSL Mode

Linearly modulated OSL capability of the OSL measurement system was tested using the same samples used in the previous tests. An example of an LM-OSL signal from $Al_2O_3$:C (irradiated with 0.5 Gy) sample is shown in figure 5.16. The stimulation light intensity is linearly increased from 0 to maximum power in a 500s interval during the measurement. As seen from the figure, the LM-OSL curve has a peak-shaped form and can be approximated using three first order LM-OSL

components (see Table 5.2). The general appearance of the LM-OSL curve and the number of components are consistent with previously reported measurement and analyses (Bulur et al., 2001). Inset to figure 5.16, LM-OSL curves of $Al_2O_3$:C samples irradiated with various doses (0-500 mGy) are also shown.

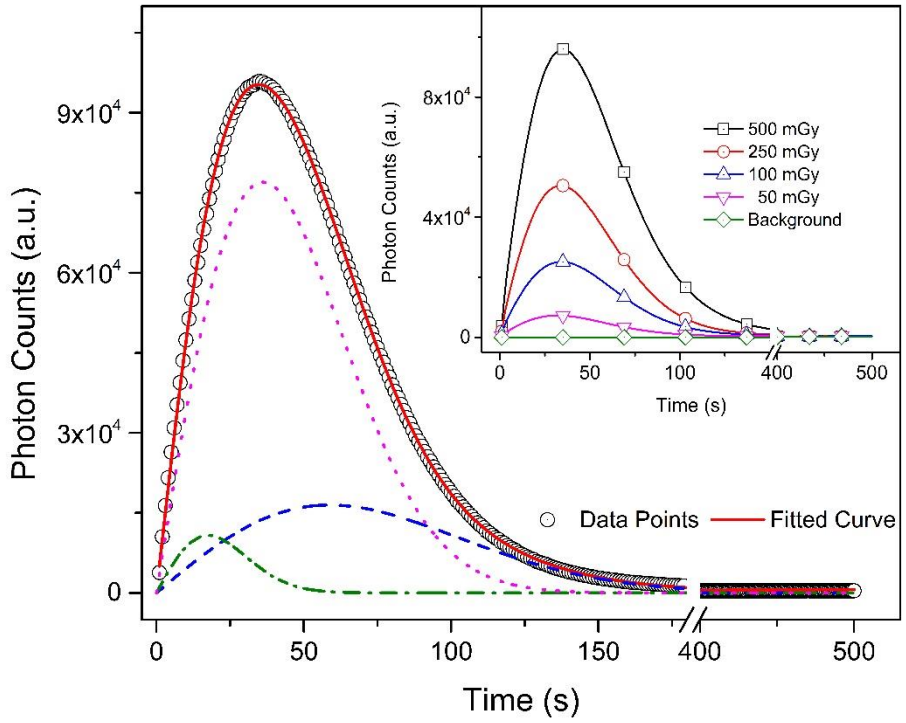



**Figure 5.16:** LM-OSL data of 0.5 Gy irradiated $Al_2O_3$:C chip with fitted curve and its components (dotted, dashed and dot-dashed lines). Inset: LM-OSL curves of $Al_2O_3$:C chips with various doses together with background.

Another material used in LM-OSL measurements was BeO. In figure 5.17, LM-OSL signal of irradiated BeO (0.5 Gy) is shown. The data was collected in 500s interval where intensity of stimulation light is linearly increased from 0 to maximum power. The data obtained from the measurement of BeO was fitted to a combination of three first order LM-OSL components given in Table 5.2 and a linearly increasing background component in a form of *ct* (where c is a contant and *t* is time).

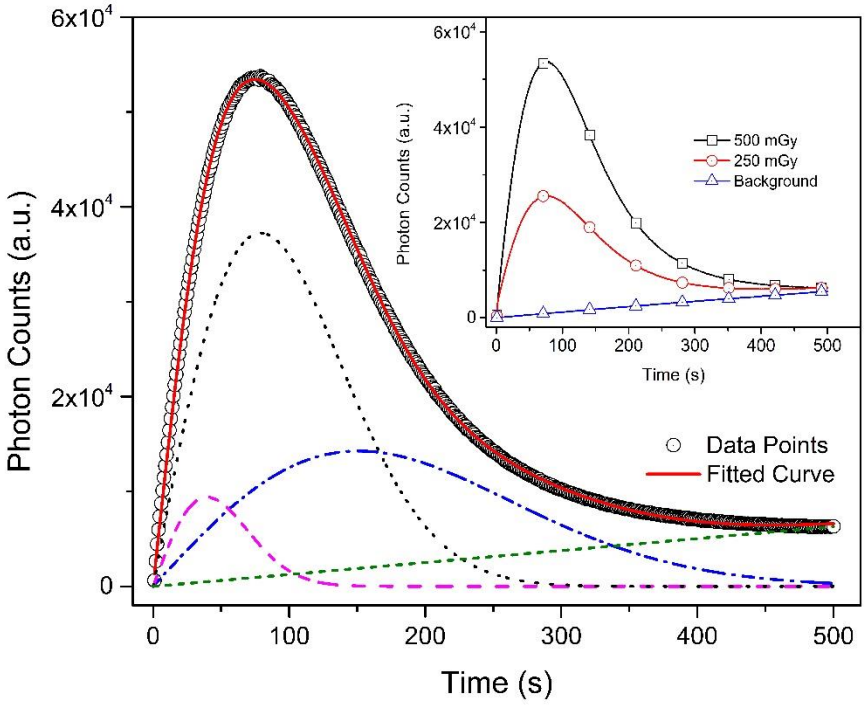



**Figure 5.17:** LM-OSL data of 0.5 Gy irradiated BeO chip with fitted curve and its components (dotted, dashed, dot-dashed and short dashed). Inset: LM-OSL curves of BeO chips irradiated at 250 mGy and 500 mGy together with measurement background.

The components of the fitted curve are given in the figure 5.17. Moreover, LM-OSL curves of BeO chips irradiated to 250 and 500 mGy is given inset to figure 5.17 with the background. Here, background signal is a repetition of the same measurement ($2^{nd}$ measurement) on 500 mGy irradiated sample. As it can be seen from the plot, there is a background linearly enhancing with increasing stimulation intensity. The reason of observing such a background is probably that not all electrons in traps were depleted after first LM-OSL measurement. It was also reported that LM-OSL signal of BeO chips can be described of combination of three curves (Bulur et. al, 2001). It is also possible to describe the obtained luminescence signal as a combination of two LM-OSL curves to conduct a relationship between CW-OSL

(see Bulur, 2010). However, this discussion is not in the scope of this study. The measurement results were consistent with measurements reported in the literature. Therefore, it can be stated that LM-OSL measurements for BeO chips are possible to perform using the automated OSL reader.

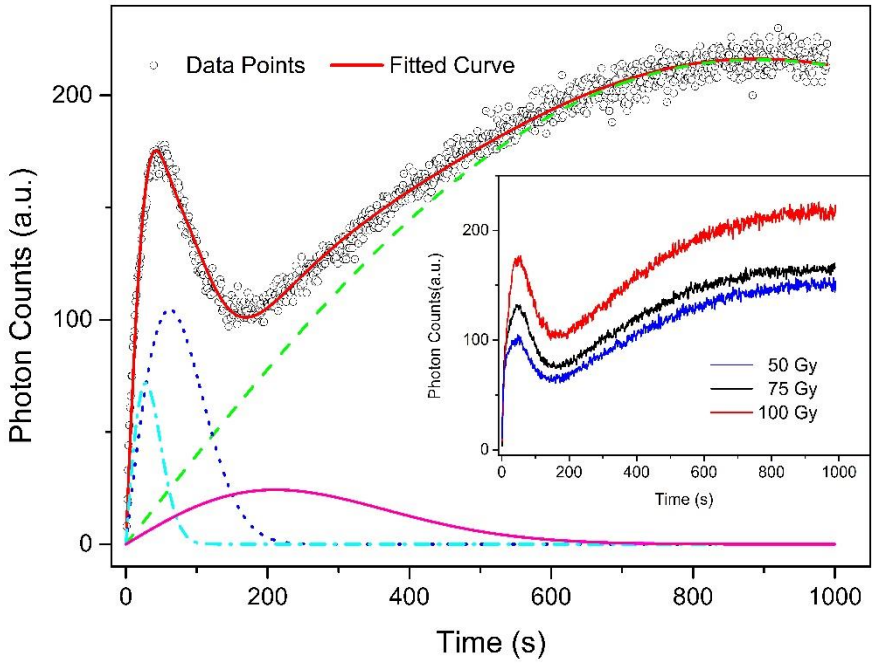



**Figure 5.18:** LM-OSL data of 100 Gy irradiated quartz grains with fitted curve and its components (dotted/blue, dashed/green, solid/magenta and dot-dashed/cyan lines). Inset: LM-OSL curves of quartz grains irradiated with 50 Gy, 75 Gy and 100 Gy.

LM-OSL test measurements was conducted on quartz samples as well. In figure 5.18, LM-OSL signal of irradiated quartz (100 Gy) is shown. Intensity of stimulation light is changed linearly from 0 to maximum power in 1000s interval during the measurement. The LM-OSL curve of quartz can be approximated using four first order LM-OSL components (see Table 5.2). The components of the

approximated curve are also given in the figure 5.18. Moreover, LM-OSL curves of quartz samples for doses 50, 75 and 100 Gy can be seen inset to figure 5.18. The enhancement of the OSL signal was observed with increasing radiation dose.

It is possible to fit LM-OSL curve of a quartz sample as a combination of multiple curves (see e.g. Bulur, 2000). The variety of LM-OSL curves shapes of samples, that are collected from different places, was also reported (Li and Li, 2006; Jain et. al., 2003). In addition, Jain et. al (2003) reported that fast components observed in the LM-OSL decay cannot be erased unless samples were pre-heated at 260°C. It has been also stated that medium and slow components cannot be removed applying pre-heating alone. It can be deduced from LM-OSL data given in figure 5.18 that observed components were ultra-fast, fast, medium and slow components remaining after pre-heating at 100°C. Hence, it may be said that LM-OSL measurements performed on quartz grains show consistency with literature.

Finally, some LM-OSL measurements were conducted using zircon samples. In figure 5.19, a plot of LM-OSL data of irradiated zircon (10 Gy) can be seen. Intensity of stimulation light changes linearly from 0 to maximum power in 1000s interval during the measurement. The peak-shaped LM-OSL curve of zircon was approximated to fit three first order LM-OSL components (see Table 5.2). The components of the fitted curve are given in the figure 5.19. It is clear that more study should be performed for investigation of LM-OSL curves of zircon sample. Moreover, LM-OSL curves of quartz samples for doses 5, 10 and 25 Gy can be seen inset to the figure 5.19. An increasing LM-OSL signal with radiation dose can be observed.

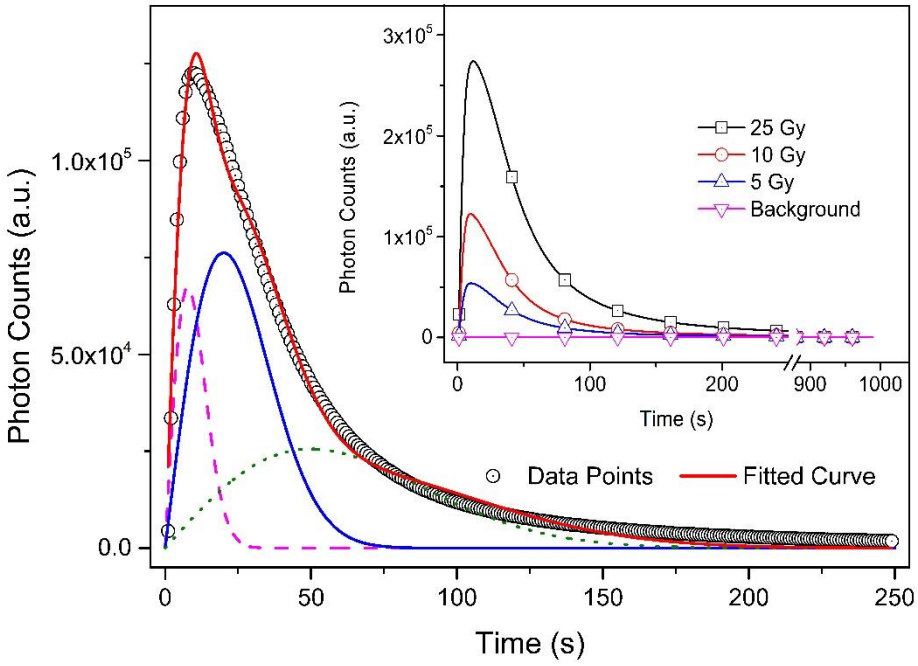



**Figure 5.19:** LM-OSL data of 10 Gy irradiated zircon sample with fitted curve and its components (solid/blue, dotted/green and dashed/magenta lines). Inset: LM-OSL curves of zircon samples irradiated with 5 Gy, 10 Gy, 25 Gy and background measurement.

**TR-OSL Mode**

TR-OSL measurements were also conducted on $Al_2O_3$:C chips for operation testing purposes. For these measurements, LED duration was set to 50 ms and luminescence data was recorded every 1 ms. The measurement accumulation was 1000 times. TR-OSL decay from 0.5 Gy irradiated $Al_2O_3$:C is given in figure 5.20. The luminescence data after stimulation was fitted into an exponential decay function given in Table 5.2. As a result of fitting, lifetime of the decay was found as around 36 ms. This calculated lifetime value is compared well with the result obtained by Markey et al. (1995), Akselrod and McKeever (1999), Pagonis et al.

(2009). Moreover, dose response measurements were conducted using TR-OSL measurement mode in the dose range between 0.1 Gy to 1 Gy. Signal intensities (obtained by deconvolution using curve-fitting) were acquired using three measurements per radiation dose step.

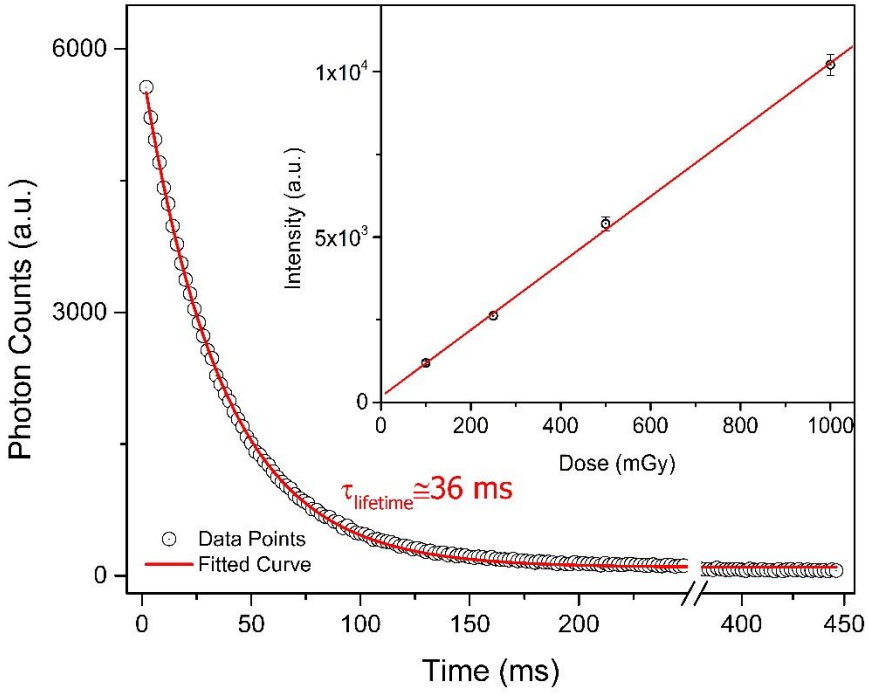



**Figure 5.20:** TR-OSL data of 500 mGy irradiated $Al_2O_3$:C chip with fitted curve. Inset: Dose response of TR-OSL measurements of $Al_2O_3$:C.

# CHAPTER 6

## SUMMARY AND CONCLUSIONS

In the scope of this thesis, a low-cost automated OSL measurement system was designed and developed utilizing open hardware and software. This multi-sample measurement device (which can be loaded up to eight samples) is able to perform three main luminescence measurement modes known as CW-, LM- and TR-OSL. It has an automated sample changer unit which consist of one dimensional tray and a rotary stage fixed on it. The tray mechanism can be taken out for loading samples with the help of a DC motor. The sample to be measured is positioned by stepper motor driven rotary stage.

The measurement chamber of the device was designed for stimulating with visible light and detecting luminescence emission in the near UV-region of the electromagnetic spectrum. For this reason, an epi-illumination measurement chamber was constructed with the help of a dichroic mirror (which has visible transmission and UV reflection) and proper filters. Stimulation of the samples is done by a blue LED ($\lambda_{peak}$ ~475 nm). A PMT module working in photon counting mode was utilized for OSL detection.

The control of sample positioning and measurements are handled by Arduino DUE microcontroller board. Moreover, a user-friendly measurement control software was written in *Python* language so that users can set measurement parameters and sequences using a computer. It is possible to set parameters like light intensity (up to ~90 mW), data collection interval (in the range from 100 μs to 5 s) and number of data points to be recorded using the measurement control software.

The communication between the computer and the instrument is maintained by serial RS-232 protocol using a USB cable.

After completion of the OSL measurement device, some experiments were conducted for determining the characteristics of stimulation LED and measurement chamber. The stability of LED temperature and output power with varying current was measured as well as time response of the LED. It was found that stimulating LED is stable in the acceptable range for the intended OSL measurements. Furthermore, background measurements showed that 45 counts per second with standard deviation of 2.30 was recorded by measurement system when the stimulation light was on (average of 12000 data points). Measured photon counts can be described using Poisson distribution since events randomly occur in a fixed time interval (Knoll, 2000). Error of Poisson measurement can be approximated as $\sigma \approx \sqrt{N}$, where $N$ is number of counts per fixed measurement time interval. Hence, the standard deviation of the background measurement was found to be lower than maximum statistical error of the measurement. Then, it can be said that precision of measurement system (including both PMT and LED source) was high and adequate for OSL measurement.

Since separation of stimulation light and luminescence emission is very important for OSL measurements, transmission and reflection characteristics of filters were measured and reported. As a result of transmission and reflection measurements, the design of measurement chamber (visible stimulation – UV detection) is verified to be suitable for OSL measurements.

Furthermore, OSL measurements were conducted on $Al_2O_3$:C chips, BeO chips, quartz grains and natural zircon grains using the constructed OSL reader. The measurement results were found to be consistent with previously reported studies in the literature. Since, there is no detailed study on CW-OSL and LM-OSL curves of natural zircon so far, presented work in this thesis related to this material could not been compared to others. However, it is overt that more work should be done with the material in order to make certain conclusions. Dose response of materials were also presented with the result of test experiments. As previously reported, the

measurements repeated three times for each dose step. Taking average of these values, the measurement uncertainty (including sample variations) was found to be around 1.41 %. Furthermore, as a result of background measurements and OSL signal of $Al_2O_3$:C and BeO chips, MDD of the instrument and dosimeters (calculated using 3σ method) was found to be less than 25 μGy. However, OSL measurements have not been conducted with the materials irradiated at these dose levels due to absence of low activity radiation source.

Test experiments with different dosimetric materials revealed that precise, reliable and repetitive measurements can be handled using this constructed automated OSL reader. However, there is no valuable measurement or information regarding the accuracy of the system. Since, neither sample variations were considered during measurements (no proper sample calibration was performed) nor samples were irradiated with a different radiation sources calibrated for each material.

In summary, the OSL measurement device, which shows its significance by having open source software and hardware utilization, was proven to be trustworthy for multi-sample OSL measurements. Even though this device has less abilities than commercial measurement devices (i.e. less sample capacity, lack of sample heating and built-in radiation source), its design using open source environment is a great advantage for experienced users/researches. It is possible for them to add new features or modify existing ones easily as necessary. Furthermore, the cost of the device (costing less than $10,000) is much less than commercial ones.

Due to the fact that this device is a part of open source family, it is and will be always possible to enhance the capabilities of the measurement system. Multi-wavelength stimulation and higher time resolution may be considered as possible features to be added soon.

## REFERENCES


Afouxenidis, D., Stefanaki, E., Polymeris, G., Sakalis, A., Tsirliganis, N., Kitis, G., 2007. TL/OSL properties of natural schist for archaeological dating and retrospective dosimetry. *Nucl Instrum Methods Phys Res A* **580**, 705–709. doi:10.1016/j.nima.2007.05.142

Aguirre, J., Alvarez, P., Followill, D., Ibbott, G., Amador, C. and Tailor, R., 2009. SU-FF-T-306: Optically Stimulated Light Dosimetry: Commissioning of An Optically Stimulated Luminescence (OSL) System for Remote Dosimetry Audits, the Radiological Physics Center *Experience. Med. Phys.* **36**, 2591-2592.

Akselrod, M., McKeever, S.W.S., 1999. A Radiation Dosimetry Method Using Pulsed Optically Stimulated Luminescence. *Radiat. Prot. Dosim.* **81**, 167–175. doi:10.1093/oxfordjournals.rpd.a032583.

Antonov-Romanovskii V. V., Keirum-Marcus I. F., Poroshina M. S. and Trapeznikova Z. A., 1956. *Conference of the Academy of Sciences of the USSR on the Peaceful Uses of Atomic Energy*, Moscow, 1955, USAEC Report AEC-tr-2435 (Pt. 1) 239.

Ashcroft, N.W., Mermin, N.D., 1976. Solid State Physics. Saunders College, Fort Worth.

Berger, T., Przybyla, B., Matthiä, D., Reitz, G., Burmeister, S., Labrenz, J., Bilski, P., Horwacik, T., Twardak, A., Hajek, M., Fugger, M., Hofstätter, C., Sihver, L., Palfalvi, J.K., Szabo, J., Stradi, A., Ambrozova, I., Kubancak, J., Brabcova, K.P., Vanhavere, F., Cauwels, V., Hoey, O.V., Schoonjans, W., Parisi, A., Gaza, R., Semones, E., Yukihara, E.G., Benton, E.R., Doull, B.A., Uchihori, Y., Kodaira, S., Kitamura, H., Boehme, M., 2016. DOSIS & DOSIS 3D: long-

term dose monitoring onboard the Columbus Laboratory of the International Space Station (ISS). *J. Space Weather Space Clim.* **6**.

Bergmann, N.W., Wallace, M., Calia, E., 2010. Low cost prototyping system for sensor networks. *2010 Sixth International Conference on Intelligent Sensors, Sensor Networks and Information Processing*. doi:10.1109/issnip.2010.5706802

Bhuyan, M., 2011. Intelligent instrumentation: principles and applications. CRC Press, Boca Raton, FL.

Bolton, W., 2015. Instrumentation and control systems. Elsevier/Newnes, Amsterdam.

Borcherds, P., 2007. Python: a language for computational physics. *Comput Phys Commun.* **177**, 199–201. doi:10.1016/j.cpc.2007.02.019.

Bøtter-Jensen, L., Larsen, N.A., Markey, B., McKeever, S.W.S, 1997. $Al_2O_3$:C as a sensitive OSL dosemeter for rapid assessment of environmental photon dose rates. *Radiat. Meas.* **27**, 295–298. doi:10.1016/s1350-4487(96)00124-2

Bøtter-Jensen, L., Bulur, E., Duller, G.A.T., Murray, A., 2000. Advances in luminescence instrument systems. *Radiat. Meas.* **32**, 523–528. doi:10.1016/s1350-4487(00)00039-1

Bøtter-Jensen, L., McKeever, S.W.S., Wintle, A.G., 2003. Optically stimulated luminescence dosimetry. Elsevier, Amsterdam.

Bräunlich P., Schäfer D., Scharmann A., 1967. A simple model for thermoluminescence and thermally stimulated conductivity of inorganic photoconducting phosphors and experiments pertaining to infra-red stimulated luminescence, *Proceedings of the First 106 International Conference on Luminescence Dosimetry*, June 1965, USAEC 57-73.

Bretthauer, D.W., 2002. Open source software: a history. *Inf. Technol. Libr* **21**, 3–10.

Bri, D., Coll, H., Garcia, M., Lloret, J., 2008. A Multisensor Proposal for Wireless Sensor Networks. *2008 Second International Conference on Sensor Technologies and Applications (sensorcomm 2008).* doi:10.1109/sensorcomm.2008.103

Buechley, L., Eisenberg, M., 2008. The LilyPad Arduino: Toward Wearable Engineering for Everyone. *IEEE Pervasive Computing* **7**, 12–15. doi:10.1109/mprv.2008.38

Bulur E., 1996. An alternative technique for optically stimulated luminescence (OSL) experiment, *Radiat. Meas*. **26**, 701-709.

Bulur E. and Göksu H.Y., 1998. OSL from BeO ceramics: new observations from an old material. *Radiat. Meas.* **29**, 639-650.

Bulur, E., Bøtter-Jensen, L., Murray, S.A., 2001. Frequency modulated pulsed stimulation in optically stimulated luminescence. *Nucl. Instrum. Meth. Phys. Res. B* **179**, 151–159.

Bulur, E., Bøtter-Jensen, L., Murray, S.A., 2001. LM-OSL signals from some insulators: an analysis of the dependency of the detrapping probability on stimulation light intensity. *Radiat. Meas.* **33**, 715–719. doi:10.1016/s1350-4487(01)00089-0

Bulur, E., Kartal, E., Saraç, B.E., 2014. Time-resolved OSL of natural zircon: A preliminary study. *Radiat. Meas.* **60**, 46–52. doi:10.1016/j.radmeas.2013.11.011

Chen R. and Leung P. L., 2002. The decay of OSL signals as stretched exponential functions, *Radiat. Meas.* **37**, 519-526.

Chithambo M.L. and Galloway R. B., 2000. On luminescence lifetimes in quartz, *Radiat. Meas.* **32**, 621-626.

Chithambo M.L., 2007. The analysis of time-resolved optically stimulated luminescence: I. Theoretical considerations, *J. Phys. D: Appl. Phys.* **40**, 1874-1879.

Choi, J.H., Kim, M.J., Cheong, C.S., Hong, D.G., 2014. Development of OSL system using two high-density blue LEDs equipped with liquid light guides. *Nucl. Instrum. Meth. Phys. Res. B* **323**, 19-24.

Cree Xlamp XQ-E LEDs, 2013. Cree Inc., viewed 01 May 2017, <http://www.cree.com/led-components/media/documents/ds-XQE.pdf>.

DiBona, C., Ockman, S., Stone, M., 1999. Opensources: voices from the Open Sources revolution. O'Reilly, Beijing.

Fisher, D.K., Gould, P.J., 2012. Open-Source Hardware Is a Low-Cost Alternative for Scientific Instrumentation and Research. *MI Modern Instrumentation* **01**, 8–20.

Gilmore, A.M., 2014. Luminescence: the instrumental key to the future of nanotechnology. Pan Stanford, Singapore.

Gordon, D., Beigl, M., Neumann, M.A., 2010. Dinam: A wireless sensor network concept and platform for rapid development. *2010 Seventh International Conference on Networked Sensing Systems (INSS)*. doi:10.1109/inss.2010.5573290

Guérin, G., Lefèvre, J.-C., 2014. A low cost TL -OSL reader dedicated to high temperature studies. *Measurement* **49**, 26e33.

Harland, L., Forster, M., 2012. Open source software in life science research: practical solutions in the pharmaceutical industry and beyond. Woodhead Publishing, Oxford.

Hughes, J.M., 2011. Real world instrumentation with Python. O'Reilly, Beijing.

Hunter, J. D., 2007. Matplotlib: A 2D Graphics Environment. *Computing in Science & Engineering* **9**, 90–95.

Huntley, D.J., Godfrey-Smith, D.I., Thewalt, M.L.W., 1985. Optical dating of sediments. *Nature* **313**, 105–107. doi:10.1038/313105a0

Hütt, G., Jaek, I., Tchonka, J., 1988. Optical dating: K-feldspars optical response stimulation spectra. *Quat. Sci. Rev.* **7**, 381–385. doi:10.1016/0277-3791(88)90033-9

Imreh, G., 2014. Python in a Physics Lab. *The Python Papers* **9:4**, pp. 1-8.

Jahn, A., Sommer, M., Ullrich, W., Wickert, M. and Henniger, J., 2013. The BeOmax system -Dosimetry using OSL of BeO for several applications. *Radiat. Meas.* **56**, 324-327.

Jain, M., Murray, S.A., Bøtter-Jensen, L., 2003. Characterisation of blue-light stimulated luminescence components in different quartz samples: implications for dose measurement. *Radiat. Meas.* **37**, 441–449. doi:10.1016/s1350-4487(03)00052-0

Karvinen, T., Karvinen, K., 2011. Make: Arduino Bots and Gadgets. O'Reilly Media.

Kavanagh, P., 2004. Open source software: implementation and management. Elsevier Digital Press, Amsterdam.

Kearfott, K.J., West, W.G., 2015. An affordable optically stimulated luminescent dosimeter reader utilizing multiple excitation wavelengths. *Appl. Radiat. Isot.* **104**, pp.87–99.

Koenka, I.J., Sáiz, J., Hauser, P.C., 2014. Instrumentino: An open-source modular Python framework for controlling Arduino based experimental instruments. *Computer Physics Communications* **185**, 2724–2729. doi:10.1016/j.cpc.2014.06.007

Knoll, G.F., 2000. Radiation detection and measurement, 3rd ed. Wiley, New York.

Krbetschek, M., Götze, J., Dietrich, A., Trautmann, T., 1997. Spectral information from minerals relevant for luminescence dating. *Radiat. Meas.* **27**, 695–748. doi:10.1016/s1350-4487(97)00223-0

Lakshmanan, A.R., 1996. Radiation induced defects and photostimulated luminescence process in BaFBr: $Eu^{2+}$. *Phys. Stat. Sol. (a)* **153**, 3–27. doi:10.1002/pssa.2211530102

Levenberg, K., 1944. A method for the solution of certain non-linear problems in least squares. *Q. J. Math* **2**, 164–168. doi:10.1090/qam/10666

Li, S.-H., Li, B., 2006. Dose measurement using the fast component of LM-OSL signals from quartz. *Radiat. Meas.* **41**, 534–541. doi:10.1016/j.radmeas.2005.04.029

Lovelock, D., Lim, S., Losasso, T., 2012. SU-C-213CD-02: The Use of Optically Stimulated Luminescent Dosimeters in a Cone Beam Quality Assurance Testing. *Medical Physics* **39**, 3604.

Lowick, S.E., Preusser, F., Wintle, A.G., 2010. Investigating quartz optically stimulated luminescence dose–response curves at high doses. *Radiat. Meas.* **45**, 975–984. doi:10.1016/j.radmeas.2010.07.010

Maraba, D., Bulur, E., 2017. Design and construction of an automated OSL reader with open source software and hardware. *Radiat. Meas.* doi:10.1016/j.radmeas.2017.04.011 (in press)

Markey, B., Colyott, L., McKeever, S.W.S., 1995. Time-resolved optically stimulated luminescence from α-Al2O3:C. *Radiat. Meas.* **24**, 457–463. doi:10.1016/1350-4487(94)001190-I

Markey, B., McKeever, S.W.S., Akselrod, M.S., Bøtter-Jensen, L., Larsen, N.A., Colyott, L., 1996. The Temperature Dependence of Optically Stimulated Luminescence From α-$Al_2O_3$:C. *Radiat. Protect. Dosim.* **65**, 185–189. doi:10.1093/oxfordjournals.rpd.a031617

Marquardt, D.W., 1963. An Algorithm for Least-Squares Estimation of Nonlinear Parameters. *Journal of the Society for Industrial and Applied* Mathematics **11**, 431–441. doi:10.1137/0111030

McKeever, S.W.S., 1985. Thermoluminescence of solids. Cambridge University Press, Cambridge.

McKeever S.W.S., Markey B.G. and Akselrod M.S., 1996. Pulsed optically stimulated luminescence dosimetry using α-$Al_2O_3$:C, *Radiat. Prot. Dosim.* **65**, 267–272.

McKeever, S.W.S., Bøtter-Jensen, L., Larsen, N.A., Duller, G., 1997. Temperature dependence of OSL decay curves: Experimental and theoretical aspects. *Radiat. Meas.* **27**, 161–170. doi:10.1016/s1350-4487(96)00106-0

McKeever, S.W.S., Chen, R., 1997. Luminescence models. *Radiat. Meas.* **27**, 625–661. doi:10.1016/s1350-4487(97)00203-5

McMahon, G., 2008. Analytical instrumentation: a guide to laboratory, portable and miniaturized instruments. J. Wiley & Sons, Chichester.

Nascimento, L., Hornos, Y., 2010. Proposal of a Brazilian accreditation program for personal dosimetry using OSL. *Radiat. Meas.* **45**, 51–59. doi:10.1016/j.radmeas.2009.11.032

Nelli, F., 2015. Python data analytics data analysis and science using Pandas, matplotlib and the Python programming language. Apress, Berkeley, CA.

Pagonis, V., Mian, S.M., Chithambo, M.L., Christensen, E., Barnold, C., 2009. Experimental and modelling study of pulsed optically stimulated luminescence in quartz, marble and beta irradiated salt. *J. Phys. D: Appl. Phys.* **42**, 055407. doi:10.1088/0022-3727/42/5/055407

Pelant, I., Valenta, J., 2016. Luminescence spectroscopy of semiconductors. Oxford University Press, Oxford.

Pearce, J., 2014. Open-source lab: how to build your own hardware and reduce research costs. Elsevier, Amsterdam.

Placko, D., 2007. Fundamentals of instrumentation and measurement. ISTE Ltd., London.

Rossum, G. van, 1995. Python tutorial, Technical Report CS-R9526, *Centrum voor Wiskunde en Informatica (CWI)*, Amsterdam

Sanborn E. N., Beard E.L., 1967. Sulfides of strontium, calcium, and magnesium in infra-red stimulated luminescence dosimetry. *Proceedings of the First International Conference on Luminescence Dosimetry, June 1965*, USAEC 183-191, Stanford

Sanderson, D., Clark, R., 1994. Pulsed photostimulated luminescence of alkali feldspars. *Radiat. Meas.* **23**, 633–639. doi:10.1016/1350-4487(94)90112-0

Sarik, J., Kymissis, I., 2010. Lab kits using the Arduino prototyping platform. *2010 IEEE Frontiers in Education Conference (FIE)*. doi:10.1109/fie.2010.5673417

Sheel, S., 2014. Instrumentation theory and applications. Alpha Science Internat., Oxford.

Sommer M. and Henniger J. 2006. Investigations of a BeO-based optically stimulated luminescence dosimeter. *Radiat. Prot. Dosim.* **119**, 394–397.

Tkachenko, N.V. 2006. Optical Spectroscopy. Amsterdam: Elsevier Science & Technology.

von Seggern, H.., 1999. Photostimulable x-ray storage phosphors: a review of present understanding. *Brazilian Journal of Physics* **29**, 254–268. doi:10.1590/s0103-97331999000200008

Whitley H. V. and McKeever S. W. S., 2000. Photoionisation of deep centers in $Al_2O_3$. *J. Appl. Phys.* **87**, 249-256.

Wintle, A., Adamiec, G., 2017. Optically stimulated luminescence signals from quartz: A review. Radiation Measurements 98, 10–33. doi:10.1016/j.radmeas.2017.02.003

Yukihara, E., Gasparian, P., Sawakuchi, G., Ruan, C., Ahmad, S., Kalavagunta, C., Clouse, W., Sahoo, N., Titt, U., 2010. Medical applications of optically

stimulated luminescence dosimeters (OSLDs). *Radiat. Meas.* **45**, 658–662. doi:10.1016/j.radmeas.2009.12.034

Yukihara, E.G., McKeever, S.W.S., 2011. Optically stimulated luminescence: fundamentals and applications. Wiley, Chichester, West Sussex.

Yukihara, E.G., Mckeever, S.W.S., Akselrod, M.S., 2014. State of art: Optically stimulated luminescence dosimetry – Frontiers of future research. *Radiat. Meas.* **71**, 15–24.

Yukihara, E., Andrade, A., Eller, S., 2016. BeO optically stimulated luminescence dosimetry using automated research readers. *Radiat. Meas.* **94**, 27–34. doi:10.1016/j.radmeas.2016.08.008

Zacharias, N., Stuhec, M., Knezevic, Z., Fountoukidis, E., Michael, C., Bassiakos, Y., 2007. Low-dose environmental dosimetry using Thermo- and Optically Stimulated Luminescence. *Nucl Instrum Methods Phys Res A* **580**, 698–701. doi:10.1016/j.nima.2007.05.125

Zhang, J., Ong, S.K., Nee, A.Y.C., 2009. Design and development of a navigation assistance system for visually impaired individuals. *Proceedings of the 3rd International Convention on Rehabilitation Engineering & Assistive Technology - ICREATE '09*. doi:10.1145/1592700.1592702

Zhou, D., Semones, E., Gaza, R., Johnson, S., Zapp, N., Weyland, M., Rutledge, R. and Lin, T., 2009. Radiation measured with different dosimeters during STS-121 space mission. *Acta Astronaut* **64**, 437-447.

# APPENDIX A

## WIRING DIAGRAMS

In this section, wiring diagrams of the designed OSL measurement system are given. The whole measurement electronics is controlled by Arduino DUE board. In figure A.1, pinout diagram of the microcontroller board is given. In table A.1, utilized input and output pins of the microcontroller is given together with their use of purpose. In the following figures, pin numbers shown in the table A.1 are referenced.

In figure A.2, wiring and connection diagram of power supplies is given. A 150W power supply (Coinstars Power Supply, Model No: KT-T150FX-06A) was used in order to supply electrical power in various voltages. This power supply has outputs of +5V (max. 8A), 3.3V (max. 5A), 12V (max. 9A), and -12V (max. 0.5A). Since there was a need for voltages like +6V and 10.5V in order to drive motors and LED, two DC-DC configurable voltage regulator modules (see http://www.instructables.com/id/The-Introduction-of-LM2596-Step-Down-Power-Module-/ for product information, accessed on June 18$^{th}$, 2017) were utilized. These modules host LM2596 (Texas Instruments) integrated circuit for voltage regulation. In order to get -5V (which is needed for PMT to run), a 3-terminal negative voltage regulator (National Semiconductor, LM7905) was used.

Figure A.3 show the wiring and connection diagram of front panel and indicators and switched. As seen from the figure, there are three LED indicators showing ‘On/Off’, ‘Busy’ and ‘Error’ status of the device. In addition, a liquid crystal display was placed on the front panel. The user can observe the current status (i.e. “in progress”, “ready”) of the device as well as current sample position using

the display. Connection of LED indicators and LCD to the microcontroller is also shown in the figure. Moreover, wiring of switches and buttons, that are placed on the front panel, are given.

The wiring and connection diagram of motors and sensors, which are used for detection of tray and rotary stage position, are shown in figure A.4. L298N (STMicroelectronics) motor driver boards were utilized for driving both stepper and DC motor. The connections of motor drivers to the microcontroller is shown in the figure. In addition to that, connections of sensor is also shown.

Finally, wiring and connection diagram of power LED (used for optical stimulation) and PMT module is given in figure A.5. Also, connections of PMT On/Off switch and its LED indicator is shown.

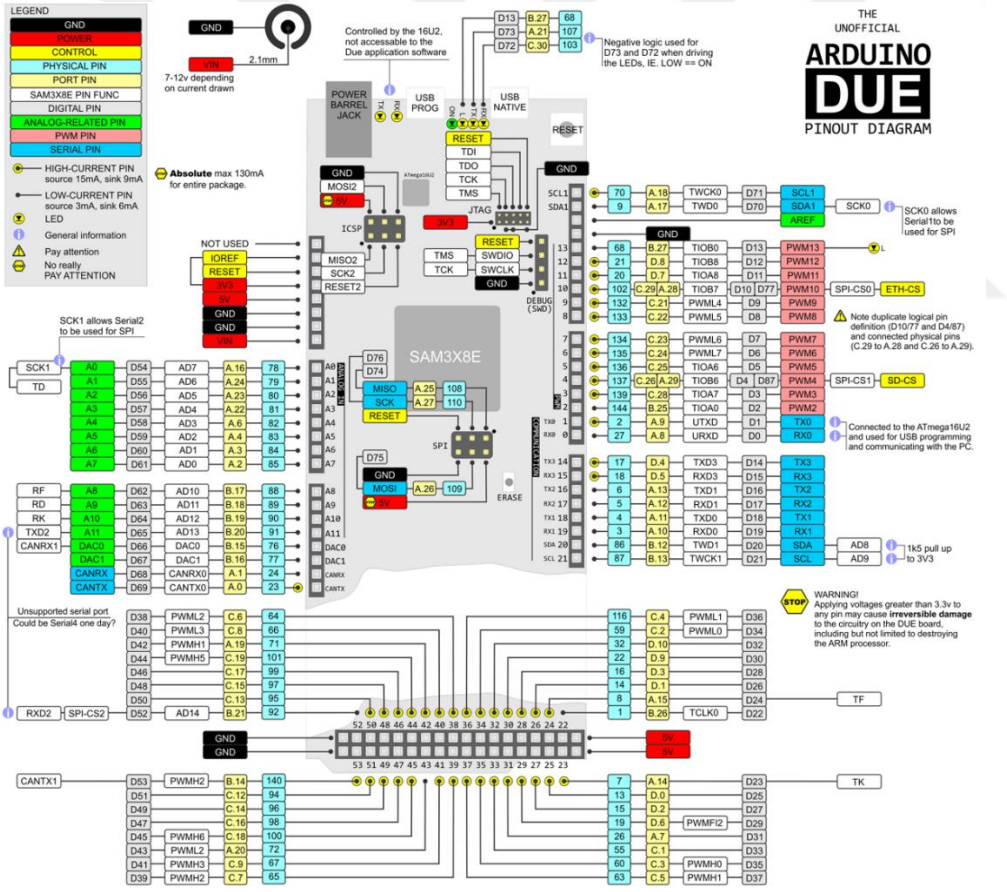



**Figure A.1:** Pinout diagram of Arduino Due microcontroller board (Taken from https://forum.arduino.cc/index.php?topic=132130.0, Accessed on June 24th, 2017).

**Table A.1:** Input and Outputs Pins of the Microcontroller

| Pin Number | Pin Mode | Use of Purpose |
|---|---|---|
| 4 | OUTPUT | DC Motor Enable |
| 6 | OUTPUT | DC Motor Control 1 |
| 7 | OUTPUT | DC Motor Control 2 |
| 8 | OUTPUT | Step Motor Control 1 |
| 9 | OUTPUT | Step Motor Control 2 |
| 10 | OUTPUT | Step Motor Control 3 |
| 11 | OUTPUT | Step Motor Control 4 |
| 12 | OUTPUT | Step Motor Driver Enable 1 |
| 13 | OUTPUT | Step Motor Driver Enable 2 |
| 22 | INPUT | PMT Output Signal (Counter Input) |
| 24 | OUTPUT | LCD Screen “RS” |
| 25 | OUTPUT | LCD Screen “Enable” |
| 34 | OUTPUT | LED Intensity (PWM) Output |
| 36 | OUTPUT | LCD Screen “Backlight” |
| 50 | OUTPUT | LCD Screen “D4” |
| 51 | OUTPUT | LCD Screen “D5” |
| 52 | OUTPUT | LCD Screen “D6” |
| 53 | OUTPUT | LCD Screen “D7” |
| A0 | INPUT | Tray Inside Sensor |
| A1 | INPUT | Tray Outside Sensor |
| A2 | INPUT PULL UP | Tray In-Switch |
| A3 | INPUT PULL UP | Tray Out-Switch |
| A4 | INPUT | Stepper Home Sensor |
| A6 | INPUT PULL UP | Next Position Switch |
| A7 | OUTPUT | Relay Control |
| A10 | INPUT PULL UP | Previous Position Switch |

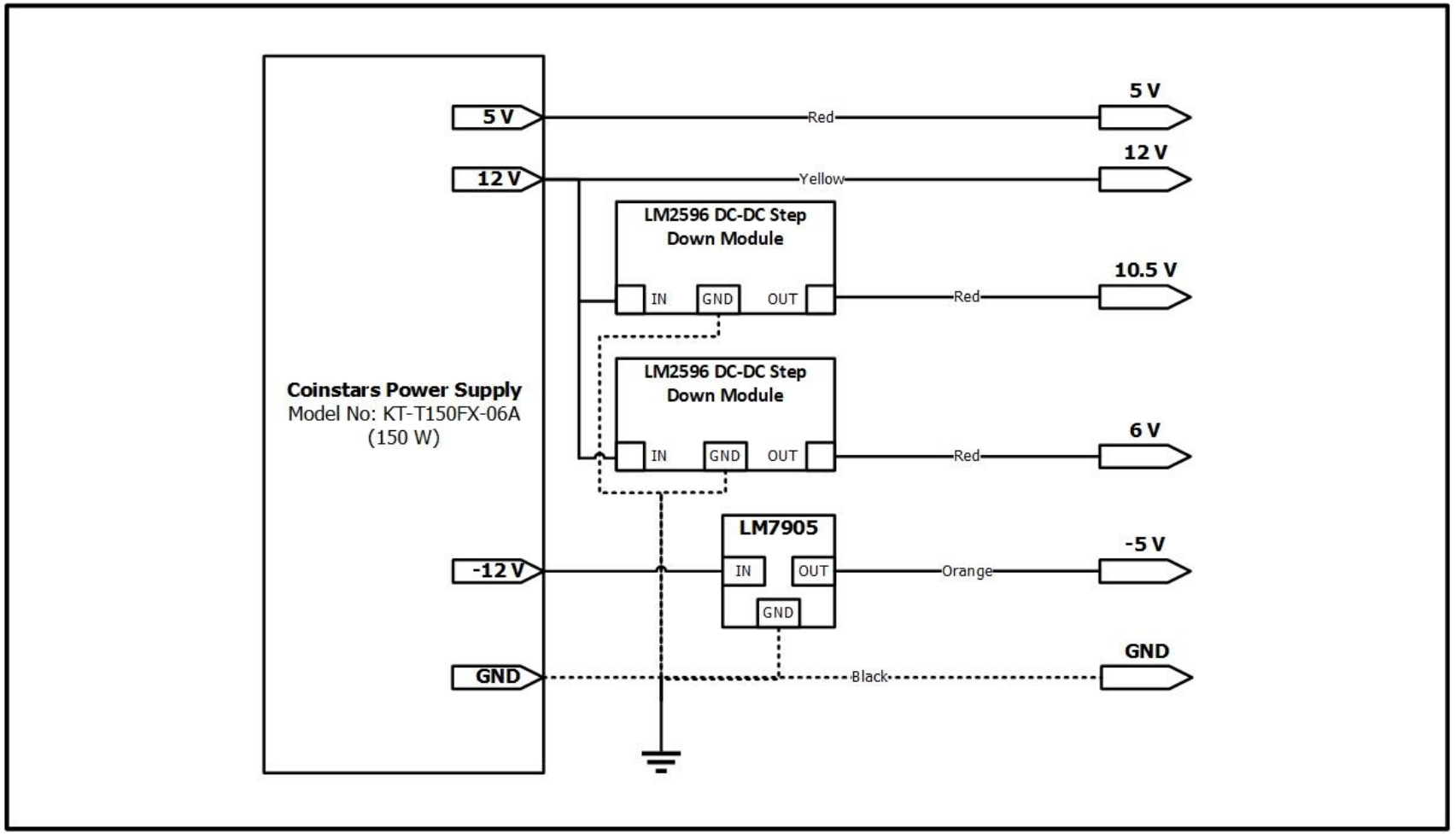



**Figure A.2:** Wiring and connection diagram of power supplies.

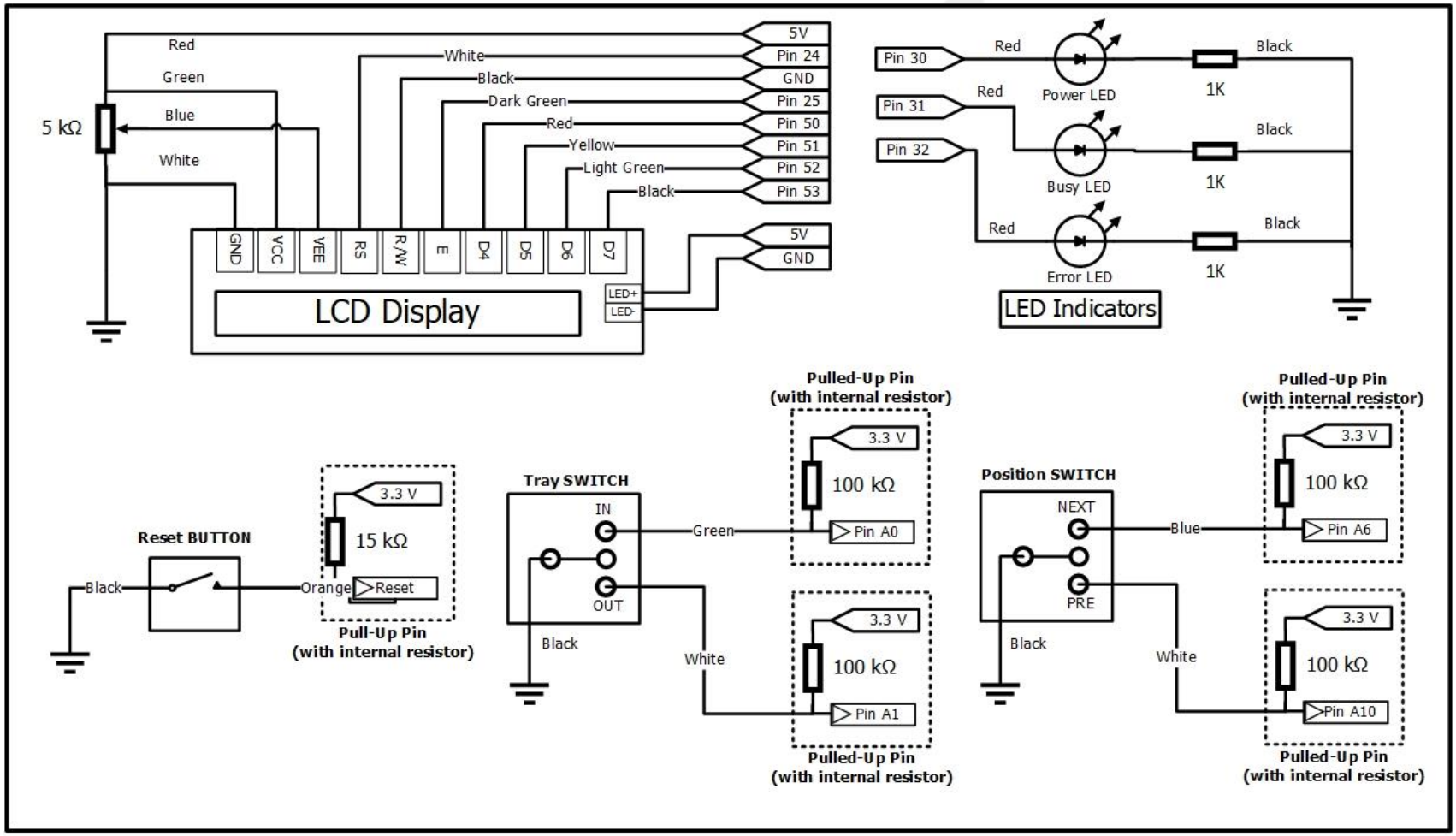



**Figure A.3:** Wiring and connection diagram of front panel indicators and switches.

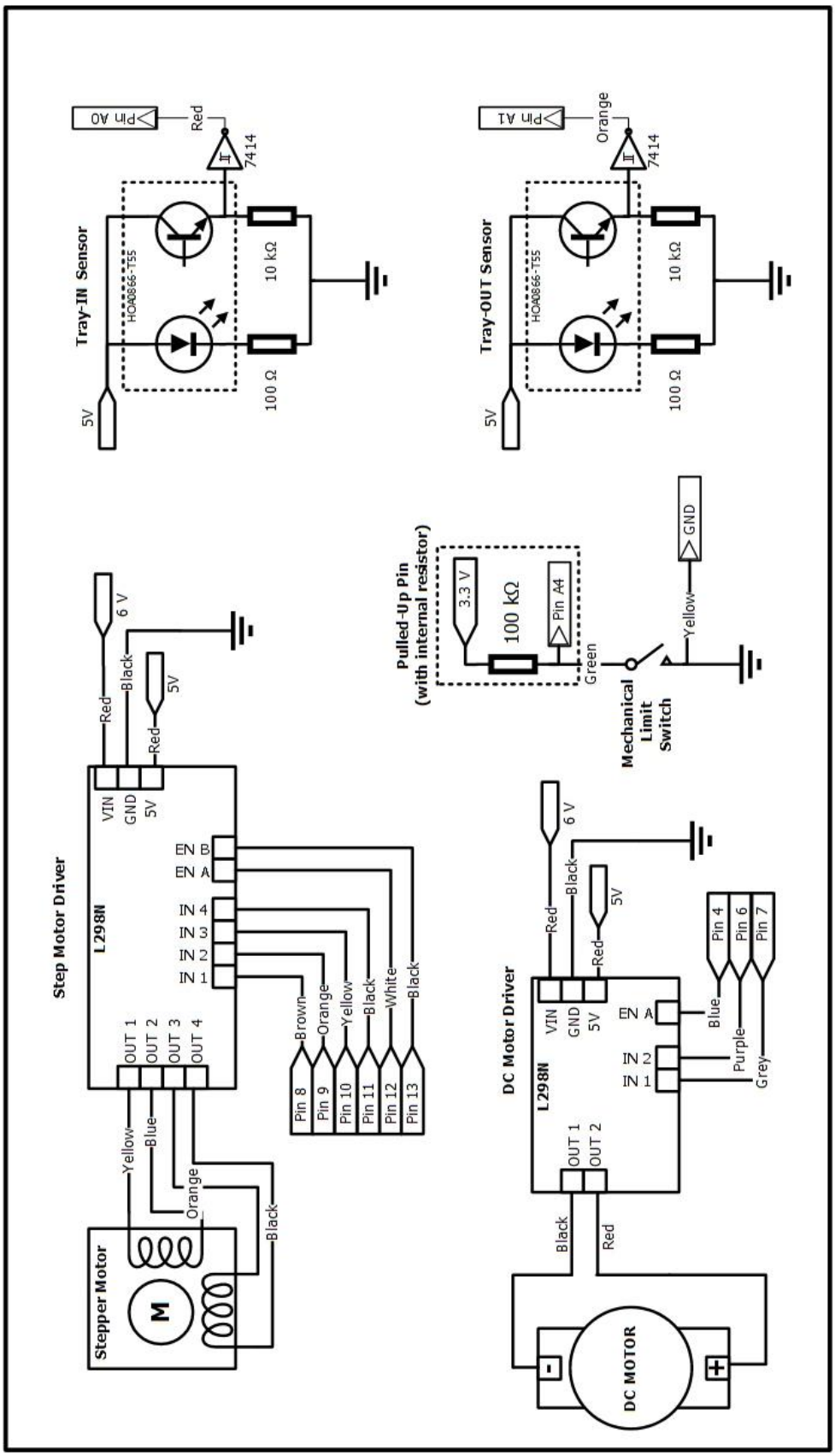



**Figure A.4:** Wiring and connection diagram of sample tray and sensor.

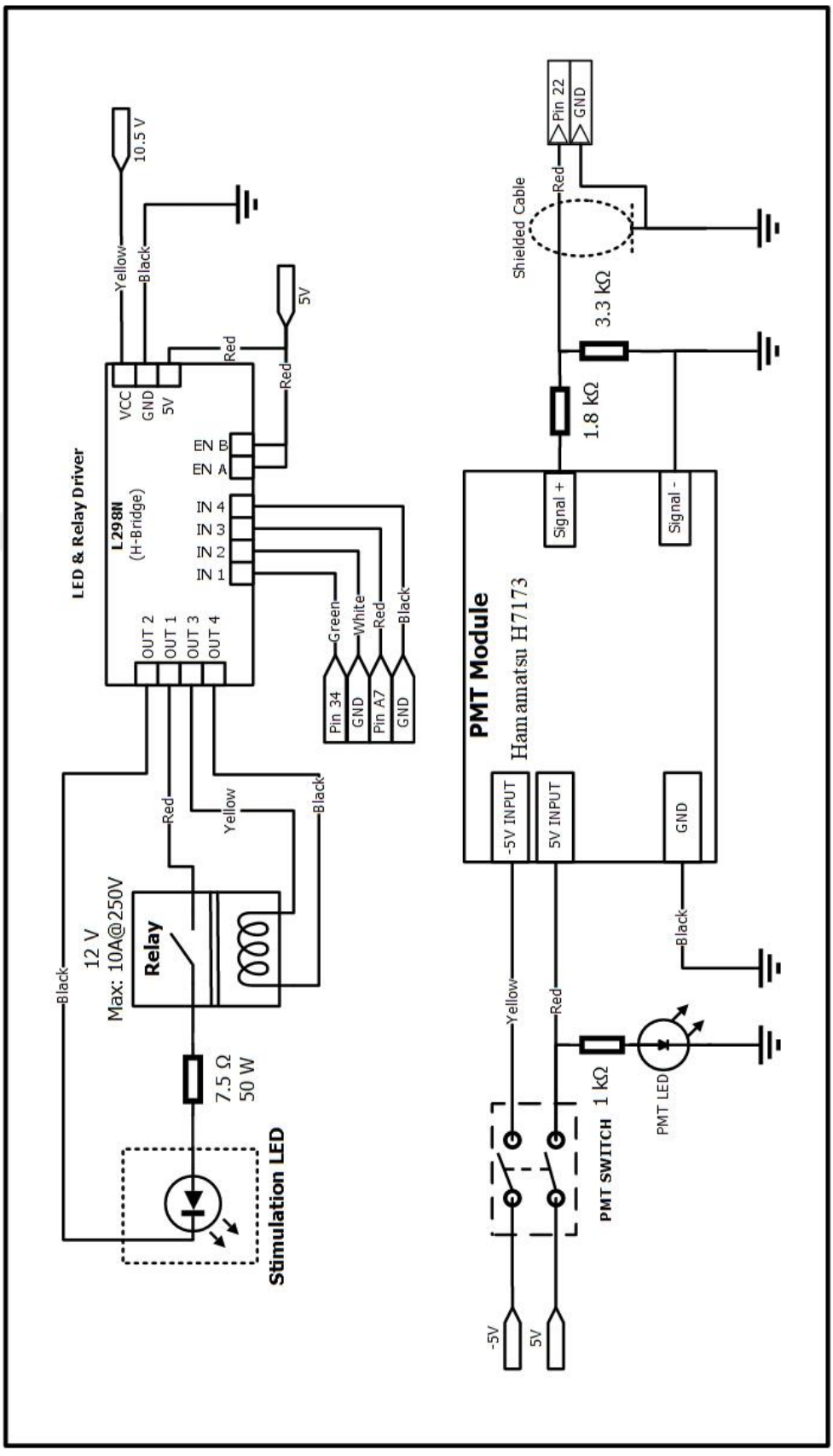



**Figure A.5:** Wiring and connection diagram of stimulation LED and PMT module.

# APPENDIX B

## GENERAL VIEW OF THE OSL MEASUREMENT SYSTEM

In this section, general view of the OSL measurement system is presented with the pictures of the components of the device.

In figure B.1, the general view of the of the OSL measurement system can be seen when sample holder tray is outside the instrument.

Figure B.2 shows a picture of measurement chamber taken when LED (OSL stimulation source) was on. PMT Module (on the right side), stimulation source (LED) and sample holder can be observed from the view.

In figure B.3, a top view of the OSL measurement system is given. Control electronics (including microcontroller, LED and motor drivers), measurement chamber, and sample holder tray can be seen together with connections

Finally, a front and rear view of the OSL measurement system is given in figure B.4 and figure B.5 respectively. In these picture, LCD screen, LED indicator (Power, Busy, Error and PMT On/Off indicators), switches (Tray in/out switch, Position switch, PMT On/Off switch and Power switch), reset button, USB Type-B connection port, D-sub connection port and 220 V power connection plug can be observed.

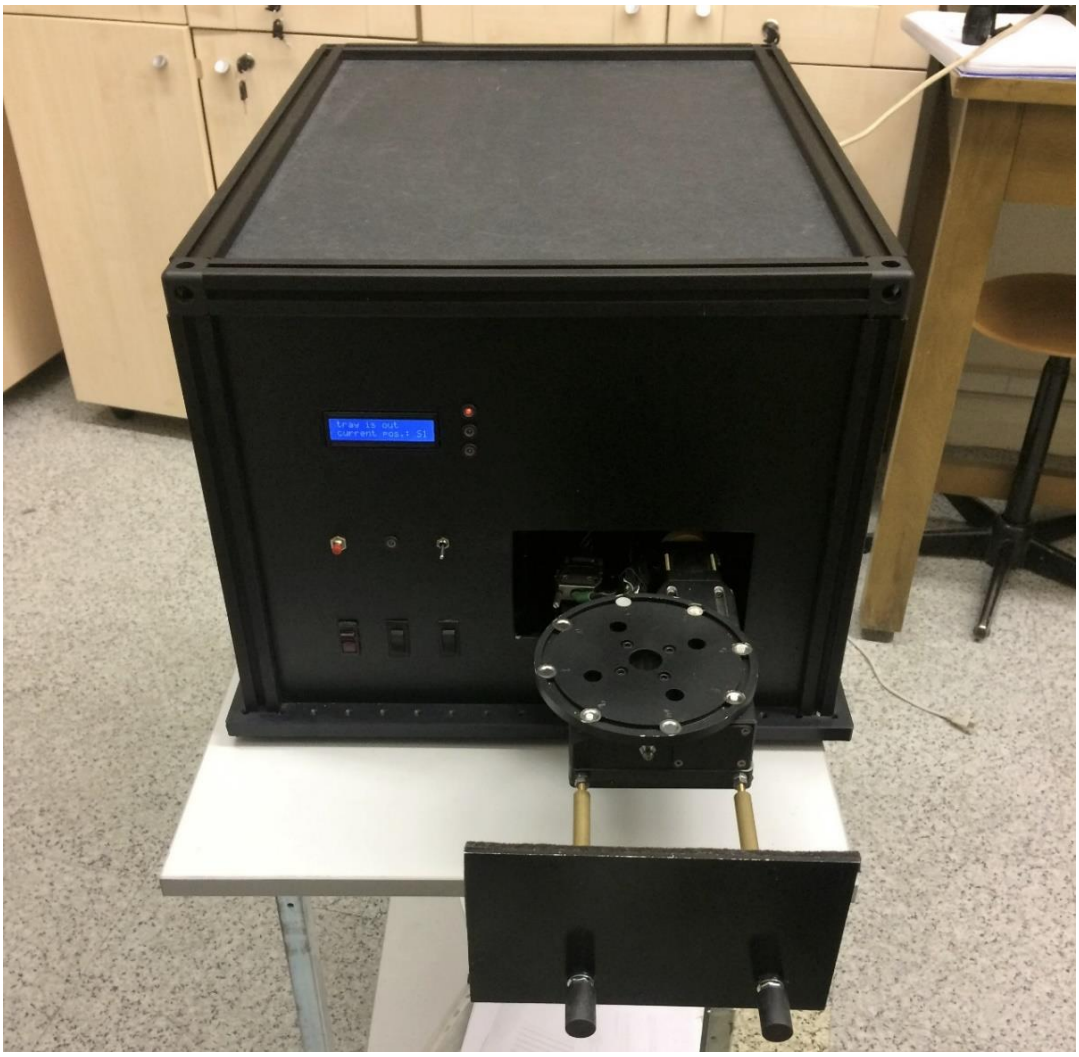



**Figure B.1:** General View of the OSL measurement system.

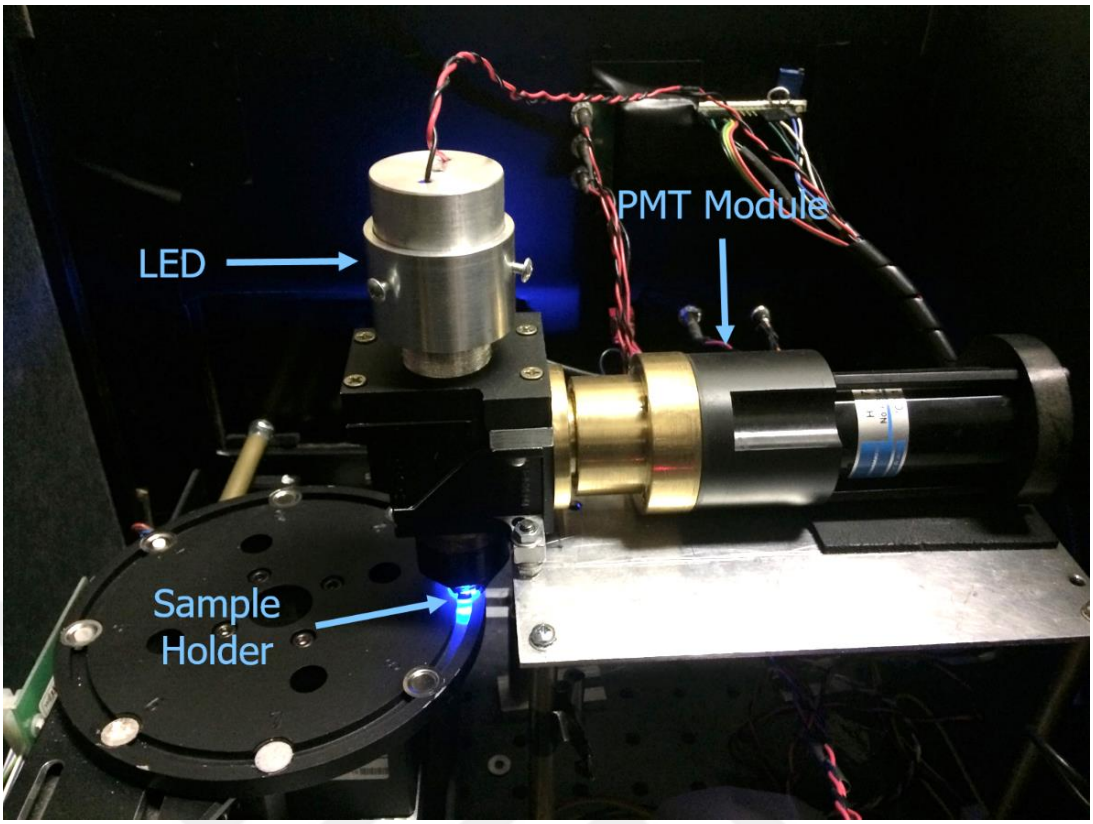



**Figure B.2:** View of Measurement Chamber. PMT Module, stimulation source (LED) and sample holder can be seen from the figure.

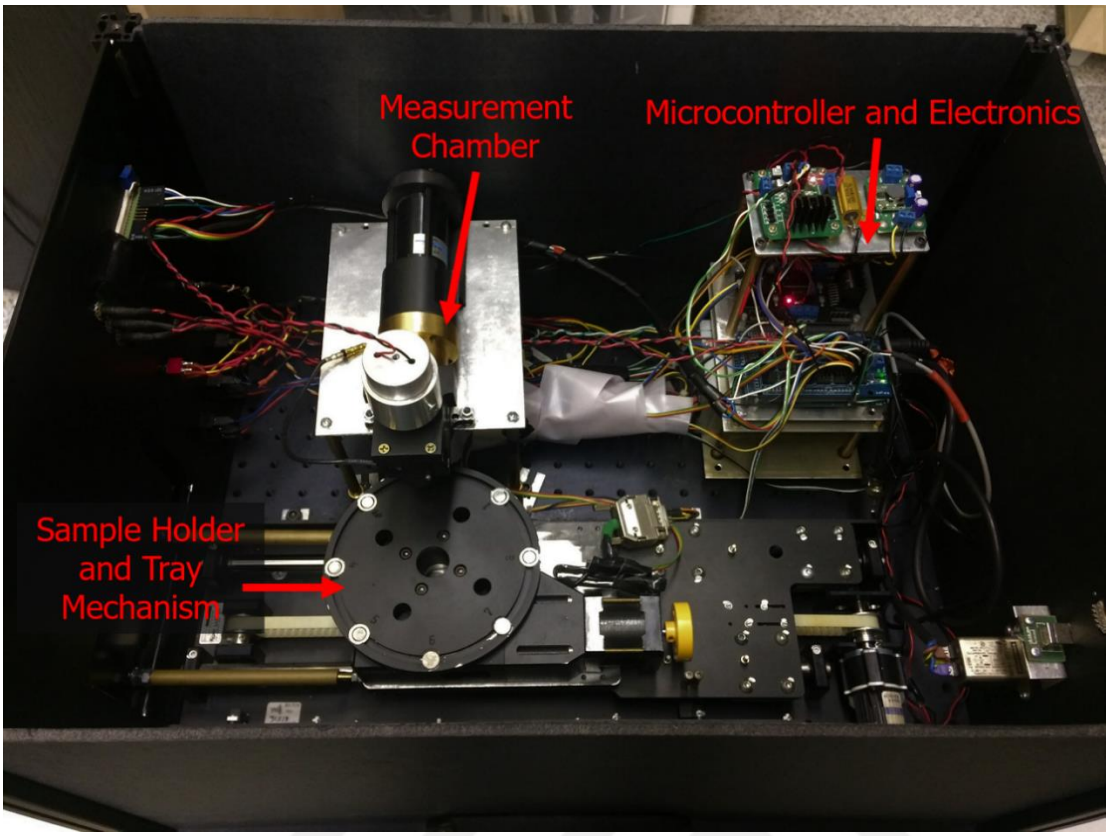



**Figure B.3:** Top view of the OSL measurement system.

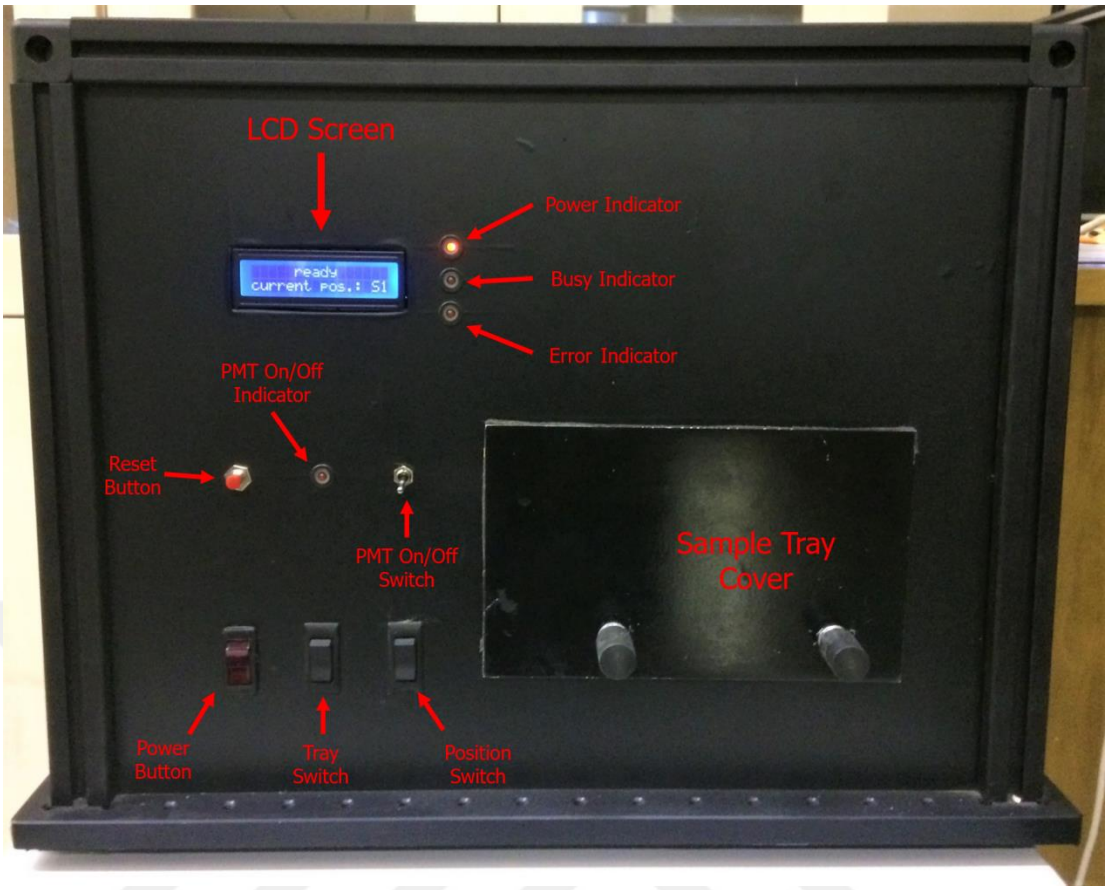



**Figure B.4:** Front view of OSL measurement system. LCD Screen, LED indicators, control switches of sample tray and PMT high voltage can be seen from the figure.

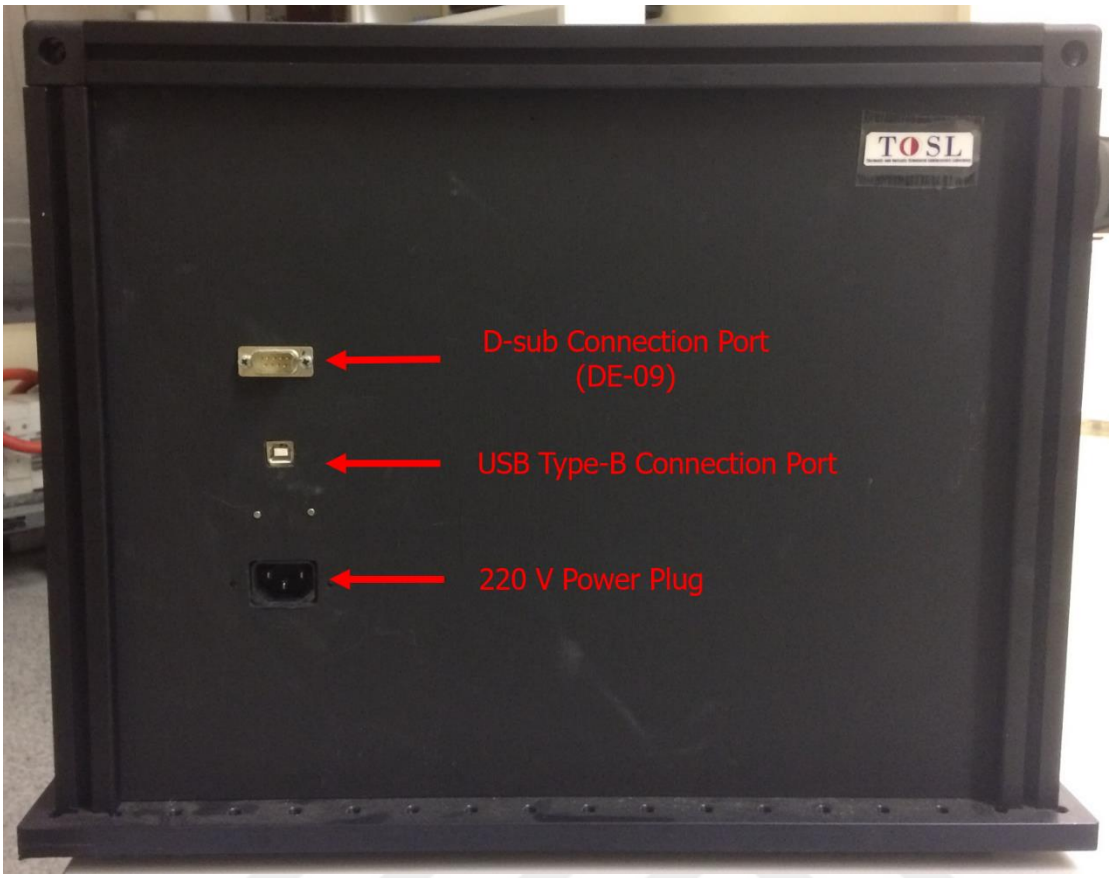



**Figure B.5:** Back view of OSL measurement system. USB connection port, D-sub port and 220 V power plug can be seen from the figure.

# APPENDIX C

## MICROCONTROLLER SOFTWARE CODE

In this section, the software code, which was compiled and uploaded to Arduino Due Microcontroller, is given together with informative comments. LiquidCrystal, Stepper and DueTimer libraries written for Arduino environment are used in the development of this code. It is possible to upload this code to any Arduino Due using Arduino IDE.

```
#include <LiquidCrystal.h>  // Liquid Crystal L-library
#include <Stepper.h>  // Stepper library
#include <DueTimer.h>   // DueTimer library by invansidel
#define TC CHANNEL 2 2   // defines timer clock channel
#define PIN_TC0_TCLK0_ARD (22u)  // defines external pin for clock
#define sei()  // defines disable global interrupt function
#define cli()  // defines enable global interrupt function
volatile long int data[20000];  // array of datas
word i = 0;  // index for data
unsigned long int k = 0;  //index for data request
boolean firstRun =0; //first run controller bit
unsigned long lastDatum= 0;  //final data on the clock register
unsigned int duty_cycle = 5000;  //duty cycle of PWM(between 0-1000)
unsigned int period = 10000;  // period of PWM (calculate using 42MHz clock =>
23.8 ns per period integer)
unsigned int ledDuration =1000000; // LED on duration time during TR-OSL
unsigned int accum = 1;  // number of accumulations (for TR-OSL mode only)
String serialData;  //defines serialData as ASCII
String param;  //defines measurement parameter input String
String inCom;  //defines incoming command string
String cwoslStr="";  //defines osl type String
String duty_cycleStr="";  //defines cw-osl duty cycle String
String idcStr="";  //defines lm-osl initial duty cycle String
String fdcStr="";  //defines lm-osl final duty cycle String
String nodStr="";  //defines number of measurement data String
String tStr="";  //defines integration time String
String ledStr=""; //defines LED duration time String
String accStr=""; //defines accumulation String
unsigned int t = 1000000;  //interval time in microseconds
unsigned int T = 100000000;  //total measurement time
float m1 = 0.00;  //before measurement time coefficent (between 0-1)
float m2 = 1.0;  //during measurement time coefficent (between 0-1)
float m3 = 0.00;  //after measurement time coefficent (between 0-1)
unsigned int n1 = T*(m1);  //before measurement time
```

```
unsigned int n2 = T*(m2);  //during measurment time
unsigned int n3 = T*(m3);  //after measurment time
int cwosl = 1;  //OSL type boolean
unsigned int countnb;  //number of measurement data
boolean readData = false;  //data send determination integer (0 or 1)
int nod = 0;  //number of datas per measurements
int idc = 0;  //initial duty cycle (0%-100%)
int fdc = 100;  //final duty cycle (0%-100%)
int idcv = (((idc)*(period))/100);  //initial duty cycle value
int fdcv = (((fdc)*(period))/100);  //final duty cycle value
int timer5 = ((n2)/(100));
int j = (idcv);  // index for duty cycle
unsigned int resolution = 1000;  //resolution of LM-OSL
int relay = A7;  // relay output pin
boolean inSensorVal;  // inSensor status bit
boolean outSensorVal;  // outSensor status bit
boolean oneTime=true;
boolean dataWrite = false;
/*************************Pin Definitions*************************/
LiquidCrystal lcd(24, 25, 50, 51, 52, 53);  //initializes LCD screen control
outputs
int backLight = 36;        //pin 36 will control the backlight
int homeSensorPin = A4;    //defines home sensor pin
int outSensorPin = A1;     //defines outside sensor pin
int inSensorPin = A0;      //defines inside sensor pin
int prePin = A10 ;         //defines backward sample switch pin
int nextPin = A6;          //defines forward sample switch pin
int inSwitch = A2;         //defines tray inside switch pin
int outSwitch = A3;        //defines tray outside switch pin
int totalStep = 0;         //defines number of steps integer
int pStep = 0;             //defines number of steps between samples integer
int currentPos = 1;        //defines current position of step motor integer
int rayPos = 0;            //defines current position of tray integer
int NOS = 8;               //defines number of samples
int lastValue = 0;
int speedstep = 150;       //defines step motor's speed
int x = 0;
int y = 0;
int z = 0;
/* Sets the home state value*/
int getHomeStateValue(){
  int value = analogRead(homeSensorPin);
  int res;
  if( lastValue >1000 && value<10){
    res = 1;
  }else{
    res = 0;
  }
  lastValue = value;
  return res;
  }
Stepper myStepper(200, 8, 9, 10, 11);  // defines stepper control output pins

/**********************Initialization Function*********************/

void setup() {
  Serial.begin(9600);// opens serial communications and sets serial data
transfer speed
  pinMode(relay, OUTPUT);// set digital pin 36 as output
  digitalWrite(relay, LOW);//disables relay
  pinMode(backLight, OUTPUT);//set pin 13 as output
  analogWrite(backLight, 100);//controls the backlight intensity
  lcd.begin(16,2);  // columns, rows. size of display
  lcd.clear();  // clear the screen

  /* Definition of Pin Types and Special Pins*/
```

```
  pinMode(inSwitch, INPUT_PULLUP);
  pinMode(outSwitch, INPUT_PULLUP);
  pinMode(nextPin, INPUT_PULLUP);
  pinMode(prePin, INPUT_PULLUP);
  pinMode(12, OUTPUT);
  pinMode(13, OUTPUT);
  pinMode(2, OUTPUT);
  pinMode(5, OUTPUT);
  pinMode(6, OUTPUT);
  pinMode(7, OUTPUT);
  pinMode(homeSensorPin, INPUT_PULLUP);
  digitalWrite(6, LOW);
  digitalWrite(7, LOW);
  myStepper.setSpeed(speedstep);  //sets stepper speed
  pinMode(PIN_TC0_TCLK0_ARD,INPUT_PULLUP);  //sets external timer/clock pin as
digital with internal pullup resistor
  cli();  // disable global interrupts
  pmc_enable_periph_clk(ID_TC2);  //enable using timer ID
  TC_Configure(TC0, TC_CHANNEL_2, TC_CMR_TCCLKS_XC0);  // configures
timer/clock, uses XC0/TCLK0 on PB26(digital pin 22)
  TC_Start(TC0, TC_CHANNEL_2);  // starts timer/clock taking external channel
as trigger
  pmc_enable_periph_clk (PWM_INTERFACE_ID) ;  // turn on clocking to PWM unit
  PWMC_ConfigureChannel (PWM, 0, 1, 0, 0) ; // PWM channel 0, clock = MCLK/2 =
42MHz (for fastest case)
  PWMC_SetPeriod (PWM, 0, period) ;  // period = number of PWM clocks
  PWMC_EnableChannel (PWM, 0) ;   // enable
  PWMC_SetDutyCycle (PWM, 0 , 0); // Configure pin 34 (PC2) to be driven by
peripheral B (PWM channel 0 L)
  PIOC->PIO_PDR = 1<<2 ;  // disable PIO control
  PIOC->PIO_IDR = 1<<2 ;   // disable PIO interrupts
  PIOC->PIO_ABSR |= 1<<2 ;  // switch to B peripheral
  sei();  // disable global interrupts
  goIn();
  digitalWrite(12, HIGH);  // enable motor driver
  digitalWrite(13, HIGH);  // enable motor driver
  delay(100);
  /* finds and sets the home state position of sample changer*/
  while(!getHomeStateValue()){
    if (i == 0){
     lcd.clear();
     lcd.print("initializing");
     i++;
   }
    else if (i!=0){
     }
    myStepper.step(100);
  }
  myStepper.step(3230);

  digitalWrite(12, LOW);  // disable motor driver
  digitalWrite(13, LOW);  // disable motor driver
  totalStep += 24000;  //total number of steps of stepper motor for a
revolution.

  pStep = totalStep/NOS;  //number of steps between consecutive sample
positions
  lcd.clear();
  lcd.setCursor(5,0);
  lcd.print("ready");
  lcd.setCursor(0,1);
  lcd.print("current pos.: S"), lcd.print(currentPos);
  delay(1000);
}
/**********************Infinite Loop Function**********************/
void loop() {
   delay(100);
```

```
 /*Gets Incoming Serial Commands*/
 if (Serial.available()){
    /*gets 2 byte long incoming commands*/
    inCom=Serial.readStringUntil('\n');
    serialData=inCom.substring(0,2);
    if(serialData == "st"){
      lcd.clear();
      lcd.setCursor(5,0);
      lcd.print("in progress");
      lcd.setCursor(0,1);
      lcd.print("current pos.: S"), lcd.print(currentPos);
      delay(10);
      if(cwosl==2){
        troslmainStart();
      }
      else{
      Start();
      }
    }
    if(serialData == "sp"){
      Stop();
    }
    if(serialData == "ti"){
      goIn();
    }
    if(serialData == "to"){
      goOut();
    }
    if(serialData == "sn"){
      sendData();
    }
    if(serialData =="rp"){
      Serial.write(rayPos);
    }
    if(serialData == "pa"){
      setParameters();
    }
    /*commands to execute when tray is inside*/
if(rayPos == 0){
    if(serialData == "p1"){
      goPosition(1);
    }
    if(serialData == "p2"){
      goPosition(2);
    }
    if(serialData == "p3"){
      goPosition(3);
    }
    if(serialData == "p4"){
      goPosition(4);
    }
    if(serialData == "p5"){
      goPosition(5);
    }
    if(serialData == "p6"){
      goPosition(6);
    }
    if(serialData == "p7"){
      goPosition(7);
    }
    if(serialData == "p8"){
      goPosition(8);
    }
    if(serialData == "pp"){
      Serial.write(currentPos);
    }
    if(serialData == "ps"){
```

```
        dataReq();
      }
    }
   }
    /*switch status commands*/
    if(digitalRead(inSwitch) == LOW){
      goIn();
    }
    if(digitalRead(outSwitch) == LOW){
      goOut();
    }
    if(rayPos != 0){
      if(digitalRead(prePin) == LOW){
        if(currentPos == 8){
          goPosition(1);
          }
        else{
        goPosition(currentPos+1);
        }
      }
      if(digitalRead(nextPin) == LOW){
        if(currentPos == 1){
          goPosition(8);
          }
        else {
        goPosition(currentPos-1);
        }
      }
    }
    /*sends measurement data to host computer*/
    if (k < (nod) && readData == true){
      delay(t/1000);
      data[-1] = 0;
      Serial.print(k+1), Serial.print("-"), Serial.print(abs(data[k]-data[k-
1]));
      Serial.print("\n");
      k++;
      if(k==nod){
        k=0;
        readData=false;
      }
     }
    if(k<(T/t) && dataWrite == true){
      for (i=0;  i < (T/t); i ++){
        Serial.print(i+1), Serial.print("-"), Serial.print(abs(data[i]-data[i-
1]));
        Serial.print("\n");
      }
      dataWrite=false;
    }
}
/*********************Counter Register Read************************/
void myHandler4 () {
  data[i]= TC_ReadCV(TC0,TC_CHANNEL_2);  // reads timer/clock and writes on
ith element of the array
  i++;
}
/****************LM-OSL LED Intensity Ramp Function***************/
void myHandler5 (){
  if (j == 0 | j < (fdcv)){
    PWMC_SetDutyCycle (PWM, 0,j);  //sets Duty Cycle
    j+=(1*((period)/(resolution)));
  }
  else if(j = (fdcv)){
    Timer1.stop();  //DueTimer.h library Timer1 stops
    j = 0;
    PWMC_SetDutyCycle (PWM, 0, 0);  //sets Duty Cycle to 0
```

```
    Timer5.stop();  //DueTimer.h library Timer5 stops
  }
}
/**********************TR-OSL Start Function**********************/
void troslmainStart(){
  digitalWrite(relay, HIGH);
  if (i >=0 |i <= 10000){  //clears data array
    for (i=0;  i < (nod+1) ; i ++){
    data[i] = 0;
    }
    firstRun=1;
    i = 0;
    readData = false;
    k = 0;
  }
  cli();
  TC0->TC_BCR = TC_BCR_SYNC;
  REG_TC0_BCR = 0x1;  //clears timer/counter TC0 register (reset counter)
  sei();
  lastDatum=(TC_ReadCV(TC0,TC_CHANNEL_2));
  for (int acm=0;  acm < (accum) ; acm ++){
   repeat();
   delayMicroseconds(T);
  }
    trsend();
  lcd.clear();
  lcd.setCursor(5,0);
  lcd.print("ready");
  lcd.setCursor(0,1);
  lcd.print("current pos.: S"), lcd.print(currentPos);
}
/********************TR-OSL Accumulation Function******************/
void repeat(){
  i=0;
  cli();
  TC0->TC_BCR = TC_BCR_SYNC;
  REG_TC0_BCR = 0x1;  //clears timer/counter TC0 register (reset counter)
  sei();
  troslStart();
  Timer4.start(t);  // starts Timer 4 using DueTimer library and sets time in
microseconds
  Timer4.attachInterrupt(myHandler4tr);  // sets myHandler function as ISR for
Timer 4 using DueTimer library
  Timer7.start(n2);  //starts Timer 7 using DueTimer library and sets time in
microseconds
  Timer7.attachInterrupt(myHandler7tr);  // sets myHandler function as ISR for
Timer 7 using DueTimer library
}
/********************TR-OSL Read Counter Register******************/
void myHandler4tr () {
  data[i]+= TC_ReadCV(TC0,TC_CHANNEL_2);  // reads timer/clock and writes on
ith element of the array
  i++;
}
/**********************TR-OSL Stop Function**********************/
void myHandler7tr(){
  Timer7.stop();
  Timer4.stop();
/******************TR-OSL Send Meas. Data Function*****************/
}
void trsend(){
  delay(10);
  for (i=0;  i < (T/t); i ++){
        Serial.print(i+1), Serial.print("-"), Serial.print(abs(data[i]-data[i-
1]));
        Serial.print("\n");
         dataWrite=false;
```

```
}
 digitalWrite(relay, LOW);
}
/************************OSL Start Function***********************/
void Start(){
  digitalWrite(relay, HIGH);
  if (i >=0 |i <= 10000){  //clears data array
    for (i=0;  i < (nod+1) ; i ++){
    data[i] = 0;
    }
    firstRun=1;
    i = 0;
    readData = false;
    k = 0;
  }
  cli();
  TC0->TC_BCR = TC_BCR_SYNC;
  REG_TC0_BCR = 0x1;  //clears timer/counter TC0 register (reset counter)
  sei();
  lastDatum=(TC_ReadCV(TC0,TC_CHANNEL_2));
  Timer4.start(t);  // starts Timer 4 using DueTimer library and sets time in
microseconds
  Timer4.attachInterrupt(myHandler4);  // sets myHandler function as ISR for
Timer 4 using DueTimer library
  Timer6.start(n1);  // starts Timer 6 using DueTimer library and sets time in
microseconds
  Timer6.attachInterrupt(myHandler6);  // sets myHandler function as ISR for
Timer 6 using DueTimer library
  if(t > 10000){
    //delay(100);
    readData = true;
    }
}
/********************OSL Emergency Stop Function*******************/
void Stop(){
  digitalWrite(relay, LOW);
  Timer4.stop();  //stops Timer 4 using DueTimer library
  Timer5.stop();  //stops Timer 5 using DueTimer library
  Timer6.stop();  //stops Timer 6 using DueTimer library
  Timer7.stop();  //stops Timer 7 using DueTimer library
  Timer8.stop();  //stops Timer 8 using DueTimer library
  PWMC_SetDutyCycle (PWM, 0 , 0);
  readData = 0;
  lcd.clear();
  lcd.setCursor(5,0);
  lcd.print("ready");
  lcd.setCursor(0,1);
  lcd.print("current pos.: S"), lcd.print(currentPos);
}
/***********************CW-OSL Start Function**********************/
void cwoslStart(){
  PWMC_SetDutyCycle (PWM, 0 , duty_cycle);  // sets CW-OSL stimulation LED
intensity

}
/***********************LM-OSL Start Function**********************/
void lmoslStart(){
  Timer5.start(T/(resolution));  //starts Timer 5 using DueTimer library and
sets time in microseconds
  Timer5.attachInterrupt(myHandler5);  // sets myHandler function as ISR for
Timer 5 using DueTimer library
  }
/************************TR-OSL LED Turn ON************************/
void troslStart(){
  PWMC_SetDutyCycle (PWM, 0 , duty_cycle);  // Turns ON LED for TR-OSL
  Timer5.start(ledDuration);
  Timer5.attachInterrupt(myHandler5tr);
```

```
}
/************************TR-OSL LED Turn OFF************************/
void myHandler5tr(){
  PWMC_SetDutyCycle (PWM, 0 , 0);
  Timer5.stop();
}
/************************OSL Type Definition************************/
void myHandler6(){
  if (cwosl == 1){
    cwoslStart();
  }
  else if (cwosl == 0){
    lmoslStart();
  }
  else if (cwosl ==2){
    troslStart();
  }
  Timer6.stop();  //stops Timer 6 using DueTimer library
  Timer7.start(n2);  //starts Timer 7 using DueTimer library and sets time in
microseconds
  Timer7.attachInterrupt(myHandler7);  // sets myHandler function as ISR for
Timer 7 using DueTimer library
}
/************************CW-OSL LED Intensity************************/
void myHandler7(){
  PWMC_SetDutyCycle (PWM, 0 ,0);
  Timer5.stop();
  j = idcv;
  Timer8.start(n3);
  Timer8.attachInterrupt(myHandler8);
}
/************************Finalize OSL Measurement************************/
void myHandler8(){
  Timer4.stop();
  Timer8.stop();
  Timer7.stop();
  //readData=false;
  digitalWrite(relay, LOW);
  countnb = i;
  if (t <= 10000){
   dataReq();
   }
  lcd.clear();
  lcd.setCursor(5,0);
  lcd.print("ready");
  lcd.setCursor(0,1);
  lcd.print("current pos.: S"), lcd.print(currentPos);
}
/* Sets data request available*/
void dataReq(){
  dataWrite = true;
}
/************************Data Send Function************************/
void sendData(){
  countnb = i;
  for (i=0;  i < countnb; i ++){
    Serial.write(i+1), Serial.write("-"), Serial.print(data[i]);
    Serial.write("\n");
    delay(10);
  }
}
/************************Tray Go-In Function************************/
void goIn(){
  lcd.clear();
  digitalWrite(4, HIGH);
  while (analogRead(inSensorPin) > 10){
    if (z == 0){
```

```
      lcd.clear();
      lcd.setCursor(0,0);
      lcd.print("tray >> in");
      lcd.setCursor(0,1);
      lcd.print("current pos.: S"), lcd.print(currentPos);
      z++;
    }
     else if (z!=0){
      }
       digitalWrite(7, HIGH);
       digitalWrite(6,LOW);
       digitalWrite(4, LOW);
       delay(100);
  }
  z = 0;
  digitalWrite(4,LOW);
  digitalWrite(7, LOW);
  digitalWrite(6,LOW);
  if(rayPos == 1){
  digitalWrite(12, HIGH);
  digitalWrite(13, HIGH);
  }
  rayPos = 0;
  lcd.clear();
  lcd.setCursor(5,0);
  lcd.print("ready");
  lcd.setCursor(0,1);
  lcd.print("current pos.: S"), lcd.print(currentPos);
  digitalWrite(12, LOW);
  digitalWrite(13, LOW);
}

/***********************Tray Go-Out Function***********************/
void goOut(){
  lcd.clear();
  digitalWrite(4, HIGH);
  while (analogRead(outSensorPin) > 10){
    if (y == 0){
     lcd.clear();
     lcd.setCursor(0,0);
     lcd.print("tray >> out");
     lcd.setCursor(0,1);
     lcd.print("current pos.: S"), lcd.print(currentPos);
     y++;
    }
     else if (y!=0){
      }
       digitalWrite(6, HIGH);
       digitalWrite(7,LOW);
       digitalWrite(5, LOW);
       delay(100);
  }
  y = 0;
  delay(10);
  digitalWrite(4, LOW);
  digitalWrite(7, LOW);
  digitalWrite(6,LOW);
  if(rayPos == 0){
  digitalWrite(12, HIGH);
  digitalWrite(13, HIGH);
  }
  rayPos = 1;
  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print("tray is out");
  lcd.setCursor(0,1);
  lcd.print("current pos.: S"), lcd.print(currentPos);
```

```
  digitalWrite(12, LOW);
  digitalWrite(13, LOW);
}
/********************Sample Positioning Function*******************/
void goPosition(int val){
  lcd.clear();
  digitalWrite(12, HIGH);
  digitalWrite(13, HIGH);
  int newPos = val;                              //defines new position of
step motor due to input value
  delay(10);  //waits 10 ms
  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print("proceeding to S"), lcd.print(newPos);
  delay(100);
  int counterPos = newPos - currentPos;          //defines position difference
between new position and current position
  int absdiff = abs(counterPos);                 //defines absolute value of
position difference
  if (absdiff >= (NOS/2))
      {
        if (counterPos > 0)
        {myStepper.step((-1)*(NOS-absdiff) * pStep); //
        }
        else
        {myStepper.step((NOS-absdiff) * pStep);
        }
      }
  else
      {
        if (counterPos > 0)
        {myStepper.step((absdiff * pStep));
        }
        else
        {myStepper.step((-1)*(absdiff * pStep));
        }
  }
  currentPos = newPos;
  lcd.clear();
  lcd.setCursor(0,1);
  lcd.print("current pos.: S"), lcd.print(currentPos);
  if(rayPos == 0){
    lcd.setCursor(5,0);
    lcd.print("ready");
    }
   else if(rayPos == 1){
    lcd.setCursor(0,0);
    lcd.print("tray is out");
    }
    digitalWrite(12, LOW);
    digitalWrite(13, LOW);
    Serial.write("p");
    Serial.print(String(currentPos));

}

/****************Set Measurement Parameters Function***************/
setParameters() {

  param = Serial.readStringUntil('\n');  //gets measurement parameters from
host computer software and writes to param string.
  /* Substring generation part for next 8 (eight) lines*/
  cwoslStr=param.substring(0,1);
  duty_cycleStr=param.substring(1,4);
  idcStr=param.substring(4,7);
  fdcStr=param.substring(7,10);
  nodStr=param.substring(10,15);
```

```
  tStr=param.substring(15,25);
  ledStr=param.substring(25,32);
  accStr=param.substring(32,37);
  /*Generated subsstring are being converted to integers (next 8 lines)*/
  cwosl=cwoslStr.toInt();
  duty_cycle=duty_cycleStr.toInt();
  duty_cycle=duty_cycle*100;
  idc=idcStr.toInt();
  fdc=fdcStr.toInt();
  nod=nodStr.toInt();
  t=tStr.toInt();
  ledDuration=ledStr.toInt();
  accum=accStr.toInt();
  T=t*nod;  //total measurement time is being calculated.
  n1 = T*(m1);  //before measurement time
  n2 = T*(m2);  //during measurment time
  n3 = T*(m3);  //after measurment time
}
/*the end of the code*/
```

# APPENDIX D

## USER INTERFACE SOFTWARE CODE

In this section, the software code of user interface, which was written in *Python* 2.7, is given together with informative comments. Tkinter, numpy and matplot libraries were used in the development of this code.

```
# -*- coding: utf-8 -*-
import pdb
import Tkinter as tk
import ttk
import platform
from module2 import *  # import Comunication Port search
from Tkinter import *  # import Tkinter Library
import Tkinter, Tkconstants, tkFileDialog
from tkMessageBox import *
from tkFileDialog import askopenfilename
import numpy as np
import time
import matplotlib

from random import randint
from PIL import Image, ImageTk

matplotlib.use("TkAgg")  # it must come after importing matplotlib
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg,
NavigationToolbar2TkAgg
from matplotlib.figure import Figure
class Interface:
    # --------initializes functions----------------------------
    def __init__(self):
        self.count = 1
        self.c = 1
        self.i = 1
        self.instant = Find_Port()
        self.main_window = Tk()
        self.main_window.title('OSL Reader - TOSL-METU')
        self.main_window.iconbitmap(default='TOSL2.ico')
        self.main_window.resizable(width=False, height=False)
        self.main_window.maxsize(width=900, height=600)

        self.init()
        self.my_buffer = []
        self.pos = 1
        self.portStatus = 0
        self.stopSeq = 0
        mainloop()
```

```
    # -------------initialize main window(buttons,figures,subplots)-----------
----------------------
    def init(self):
        """ sets dropdown menus on main window """
        m = Menu(self.main_window)
        self.main_window.config(menu=m)
        self.port_menu = Menu(m)
        self.sequence_menu = Menu(m)
        self.about_menu = Menu(m)
        m.add_cascade(label="Sequence", menu=self.sequence_menu)
        m.add_cascade(label="Setup", menu=self.port_menu)
        m.add_cascade(label="About", menu=self.about_menu)
        self.sequence_menu.add_command(label="Open Sequence",
command=self.openSequence2)
        self.sequence_menu.add_command(label="Save Sequence",
command=self.saveSequence)
        self.sequence_menu.add_command(label="Add Sequence",
command=self.openSequence)
        self.about_menu.add_command(label="Credits", command=self.popCredits)
        """Puts present ports to port menu"""
        for i in self.instant.com:
            self.port_menu.add_command(label=i, command=lambda i=i:
self.port_o(i))
            # self.port_menu.entryconfig( self.port_menu.index(i),
command=self.port_o(i)
        """initializing check buttons' variables"""
        self.var1 = IntVar()
        self.var2 = IntVar()
        self.var3 = IntVar()
        self.var4 = IntVar()
        self.var4 = IntVar()
        self.var5 = IntVar()
        self.var6 = IntVar()
        self.var7 = IntVar()
        self.var8 = IntVar()

        self.showTime = time.strftime('%H:%M:%S', time.gmtime(0))
        self.remaningTime = 0.0
        tabSection = ttk.Notebook(self.main_window)
        setupTab = Label(tabSection)
        plotTab = Label(tabSection)
        tabSection.add(setupTab, text="Setup")
        tabSection.add(plotTab, text="Live Plot")
        tabSection.grid()

        frame_main = LabelFrame(setupTab, relief=FLAT)
        frame_main.grid(row=1, column=1)
        frame = LabelFrame(frame_main, bd=0, height=15)
        frame.grid(row=1, column=2, rowspan=2)
        """ Label frame of check buttons"""

        framealt = LabelFrame(frame, bd=3)
        framealt.grid()
        framealtr = LabelFrame(framealt, bd=0)
        framealtr.grid(row=1, column=2)
        framealtl = LabelFrame(framealt, bd=3)
        framealtl.grid(row=1, column=1)
        bard = Image.open("TOSL2.png")
        bardejov = ImageTk.PhotoImage(bard)
        frameimage = Label(framealtr, image=bardejov, relief=RAISED, bd=3)
        frameimage.image = bardejov
        frameimage.grid(row=0, column=1)

        """Label frame of check buttons and frame of other entry widgets"""
        frame2 = LabelFrame(setupTab, bd=2, height=15)
        frame2.grid(row=1, column=3)
```

```
        frame21 = LabelFrame(framealtl, text="OSL Type", bd=5, font="Averia 15
bold", relief=RIDGE)
        frame21.grid(row=1, column=1)
        frame22 = LabelFrame(framealtr, text="LED Intensity %", bd=0,
font="Averia 12")
        frame22.grid(row=2, column=1)
        frame26 = LabelFrame(framealtr, text="LED Duration", bd=0,
font="Averia 12")
        frame26.grid(row=3, column=1)
        frame232 = LabelFrame(framealtr, text="Final Intensity", bd=0,
font="Averia 12")
        frame232.grid(row=4, column=1)

        frame24 = LabelFrame(framealtr, text="# of Data", bd=0, font="Averia
12")
        frame24.grid(row=5, column=1)
        frame25 = LabelFrame(framealtr, text="Integration Time", bd=0,
font="Averia 12")
        frame25.grid(row=6, column=1)
        frame233 = LabelFrame(framealtr, text="Accumulation", bd=0,
font="Averia 12")
        frame233.grid(row=7, column=1)
        frameSample = Frame(framealtl, width=50, height=500, bd=5)
        frameSample.grid(row=2, column=1)
        frameButtons = Frame(setupTab)
        frameButtons.grid(row=2, column=1)
        self.statusframe = Label(frame, text="Please Select a Port",
bg="Black", fg="Red", font="Averia 12", width=26,
                                 height=2, wraplength=238)
        self.statusframe.grid(row=2, column=0, columnspan=1)

        frame4 = LabelFrame(frame_main, bd=2, height=16)
        frame4.grid(row=1, column=1)
        timeframes = LabelFrame(plotTab, bd=4, relief=RAISED)
        timeframes.grid(row=1, column=2)
        timeremain = Label(timeframes, text="Remaining Time", font="Averia 12
bold")
        timeremain.grid(row=0)
        self.clockframe1 = Label(timeframes, text=self.showTime, font="Averia
12 bold", bg="black", fg="red", width=10,
                                 height=1)
        self.clockframe1.grid(row=1, padx=70)
        """inserts  ploting area on main page """
        self.fig = Figure(figsize=(7, 5), facecolor="white")
        canvas1 = FigureCanvasTkAgg(self.fig, plotTab)
        canvas1.get_tk_widget().grid(row=1, column=0)
        self.sub = self.fig.add_subplot(111)
        self.sub.plot([], [])
        self.sub.set_ylim(0, 1000)
        self.sub.set_xlim(0, 450)

        myframe = Frame(relief=GROOVE, width=50, height=75, bd=1)
        myframe.place(x=1, y=700)

        canvas = Canvas(myframe)
        canvas.pack()
        frame = Frame(canvas)
        frame.pack()

        self.fig1 = Figure(figsize=(50, 3), facecolor="white")
        eleman = FigureCanvasTkAgg(self.fig1, frame)
        # canvas.show()
        eleman.get_tk_widget().pack()

        ######################################################################

        # def entry(self):
```

```
        """initializing check buttons' variables"""
        self.var1 = IntVar()
        self.var2 = IntVar()
        self.var3 = IntVar()
        self.var4 = IntVar()
        self.var4 = IntVar()
        self.var5 = IntVar()
        self.var6 = IntVar()
        self.var7 = IntVar()
        self.var8 = IntVar()

        # -----------check buttons sample-1 to sample-8-------------#

        """Initializing place of check buttons"""
        """Each check buttons is inside one frame"""
        self.check_1 = Checkbutton(frameSample, text="Sample 1",
variable=self.var1, command=self.enable)
        self.check_1.grid(row=1, column=1)
        self.check_2 = Checkbutton(frameSample, text="Sample 2",
variable=self.var2, command=self.enable)
        self.check_2.grid(row=2, column=1)
        self.check_3 = Checkbutton(frameSample, text="Sample 3",
variable=self.var3, command=self.enable)
        self.check_3.grid(row=3, column=1)
        self.check_4 = Checkbutton(frameSample, text="Sample 4",
variable=self.var4, command=self.enable)
        self.check_4.grid(row=4, column=1)
        self.check_5 = Checkbutton(frameSample, text="Sample 5",
variable=self.var5, command=self.enable)
        self.check_5.grid(row=5, column=1)
        self.check_6 = Checkbutton(frameSample, text="Sample 6",
variable=self.var6, command=self.enable)
        self.check_6.grid(row=6, column=1)
        self.check_7 = Checkbutton(frameSample, text="Sample 7",
variable=self.var7, command=self.enable)
        self.check_7.grid(row=7, column=1)
        self.check_8 = Checkbutton(frameSample, text="Sample 8",
variable=self.var8, command=self.enable)
        self.check_8.grid(row=8, column=1)

        # ------------------osl check buttons-----------------------#

        """initializing cw-osl and lm-osl check buttons """
        self.v1 = IntVar()
        self.v2 = IntVar()
        self.v3 = IntVar()
        self.osl_button1 = Checkbutton(frame21, variable=self.v1, text="CW-
OSL", fg="red", font="times", state="normal",
                                       command=self.cw_osl)
        self.osl_button1.grid(row=1, column=1)
        self.osl_button2 = Checkbutton(frame21, variable=self.v2, text="LM-
OSL", fg="blue", font="times",
                                       state="normal", command=self.lm_osl)
        self.osl_button2.grid(row=2, column=1)
        self.osl_button3 = Checkbutton(frame21, variable=self.v3, text="TR-
OSL", fg="Dark Green", font="times",
                                       state="normal", command=self.tr_osl)
        self.osl_button3.grid(row=3, column=1)

        # ----------------values entry buttons----------------------#

        """initializing entry boxes"""
        self.cw_osl_duty_cycle_entry1 = Entry(frame22, state="disabled",
width=25, disabledbackground="grey")
        self.cw_osl_duty_cycle_entry1.grid()
        #
self.lm_osl_initial_duty_cycle_entry21=Entry(frame231,state="disabled")
```

```
        # self.lm_osl_initial_duty_cycle_entry21.grid()
        self.lm_osl_final_duty_cycle_entry22 = Entry(frame232,
state="disabled", width=25, disabledbackground="grey")
        self.lm_osl_final_duty_cycle_entry22.grid()

        self.tr_osl_sduration_entry23 = Entry(frame26, state="disabled",
width=16, disabledbackground="grey")
        self.tr_osl_sduration_entry23.grid(row=2)
        self.number_of_data_entry = Entry(frame24, state="disabled", width=25,
disabledbackground="grey")
        self.number_of_data_entry.grid(row=3)
        self.deviation_time_entry = Entry(frame25, state="disabled", width=16,
disabledbackground="grey")
        self.deviation_time_entry.grid(row=4, column=1)
        self.integvariable = StringVar(frame25)
        self.integvariable.set("s")
        self.timemag_entry = OptionMenu(frame25, self.integvariable, "s",
"ms", "µs")
        self.timemag_entry.grid(row=4, column=2)
        self.timemag_entry.config(width=1)
        self.acc_entry = Entry(frame233, state="disabled", width=25,
disabledbackground="grey")
        self.acc_entry.grid(row=5, column=1)
        self.accvariable = StringVar(frame26)
        self.accvariable.set("s")
        self.accmag_entry = OptionMenu(frame26, self.accvariable, "s", "ms",
"µs")
        self.accmag_entry.grid(row=2, column=2)
        self.accmag_entry.config(width=1)

        # -------------------------add list buttons-----------------#
        """delete button, start button , change button"""
        self.button9 = Button(framealtr, text="Add", state="disabled",
command=self.add_list)
        self.button9.grid(row=8, column=1)

        frame41 = LabelFrame(frame4, bd=0)
        frame41.grid(row=1, column=1)
        frame4.grid_rowconfigure(0, weight=1)
        frame4.grid_columnconfigure(0, weight=1)
        self.listbox = Listbox(frame4, width=35, height=16, font="Averia",
fg="black", selectmode=EXTENDED)
        self.listbox.grid(row=1, column=2)
        self.scrollbar = Scrollbar(frame4, orient=VERTICAL)
        self.listbox.configure(yscrollcommand=self.scrollbar.set)
        self.scrollbar.config(command=self.listbox.yview)
        self.scrollbar.grid(row=1, column=3, sticky=N + S)
        self.button10 = Button(frame41, text="Delete", font="Averia 10",
width=6,
                               command=lambda: self.delete("general"))
        self.button10.grid(row=3, column=1, pady=15)

        self.button11 = Button(frame41, text="↑ Up", font="Averia 10",
width=6, command=lambda: self.change("up"))
        self.button11.grid(row=1, column=1)
        self.button11 = Button(frame41, text="↓ Down", font="Averia 10",
width=6, command=lambda: self.change("down"))
        self.button11.grid(row=2, column=1)
        frameButtons = Frame(frame_main)
        frameButtons.grid(row=2, column=1)

        self.button12 = Button(frameButtons, text="Start", fg='dark green',
font="Averia 12", command=self.start,
                               state="disabled")
        self.button12.grid(row=1, column=1)
        self.button13 = Button(frameButtons, text="Stop", font="Averia 12",
fg='dark red', command=self.stop,
```

```
                                state="disabled")
        self.button13.grid(row=1, column=2)

        self.clearAll = Button(frameButtons, text="Delete Sequence",
font="Averia 12", state="normal",
                               command=self.clearSequence)
        self.clearAll.grid(row=1, column=0, padx=6)
        self.clockframe = Label(frameButtons, text=self.showTime, font="Averia
12 bold", bg="black", fg="red", width=10,
                                height=1)
        self.clockframe.grid(row=1, column=3, columnspan=2, padx=4)

        # ------------------------------------------------------
        self.top = Toplevel()
        self.top.withdraw()
        self.top.protocol(name="WM_DELETE_WINDOW", func=self.top_level)

    def create_sub(self):
        self.sub_count = 10
        self.sub1 = self.fig1.add_subplot(1, self.sub_count, self.i)
        self.sub1.plot([], [])
        self.sub1.set_xlim(0, 200)
        self.sub1.set_ylim(0, 300)
        self.i = self.i + 1

    def top_level(self):
        self.top.withdraw()

    #--when cw osl type is checked it activates cw osl type entries-#

    """controls state of cw-osl """

    def cw_osl(self):
        self.osl_button1.select()
        self.osl_button2.deselect()
        self.osl_button3.deselect()
        self.cw_duty()

    def cw_duty(self):
        self.cw_osl_duty_cycle_entry1.config(state="normal")
        # self.lm_osl_initial_duty_cycle_entry21.config(state="disabled")
        self.lm_osl_final_duty_cycle_entry22.config(state="disabled")
        self.number_of_data_entry.config(state="normal")
        self.deviation_time_entry.config(state="normal")
        self.tr_osl_sduration_entry23.config(state="disabled")
        self.acc_entry.config(state="disabled")
        self.enable()

     # ------when lm osl type is checked it activates lm osl type entries-----
     ------------------------------------------------#

    def lm_osl(self):
        self.osl_button2.select()
        self.osl_button1.deselect()
        self.osl_button3.deselect()
        self.lm_duty()

    def lm_duty(self):
        self.cw_osl_duty_cycle_entry1.config(state="normal")
        # self.lm_osl_initial_duty_cycle_entry21.config(state="normal")
        self.lm_osl_final_duty_cycle_entry22.config(state="normal")
        self.number_of_data_entry.config(state="normal")
        self.deviation_time_entry.config(state="normal")
        self.tr_osl_sduration_entry23.config(state="disabled")
        self.acc_entry.config(state="disabled")
        self.enable()
```

```
    #--when tr osl type is checked it activates tr osl type entries-#

    def tr_osl(self):
        self.osl_button3.select()
        self.osl_button1.deselect()
        self.osl_button2.deselect()
        self.tr_duty()

    def tr_duty(self):
        self.cw_osl_duty_cycle_entry1.config(state="disabled")
        # self.lm_osl_initial_duty_cycle_entry21.config(state="disabled")
        self.lm_osl_final_duty_cycle_entry22.config(state="disabled")
        self.number_of_data_entry.config(state="normal")
        self.deviation_time_entry.config(state="normal")
        self.tr_osl_sduration_entry23.config(state="normal")
        self.acc_entry.config(state="normal")
        self.enable()

     # ------when checked samples this function changes add list button's
     satate to normal----------------------------------#
    # def button able(self):
    # self.button9.config(state="normal")

     # ------controls whether the samples checked or not also changes state of
     add list button-----------------------------------#
    def enable(self):
        """if sample checked keeps 1 otherwise keeps 0 """
        self.sample_array = [self.var1.get(), self.var2.get(),
self.var3.get(), self.var4.get(), self.var5.get(),
                             self.var6.get(), self.var7.get(),
self.var8.get()]
        count = self.sample_array.count(1)
        if ((count == 0) or (self.v1.get() == 0 and self.v2.get() == 0) and
self.v3.get() == 0):
            """if  samples are not selected or osl types are not selected, it
disables add-list button. Otherwise it anables"""
            self.button9.config(state="disabled")
        else:
            self.button9.config(state="normal")

    # ------gets values from entries then sends these values "add" function
which appends values to the list box------------#
    def popCredits (self):
        showinfo("Credits", "Designed by Diren Maraba in Thermally and
Optically Stimulated Luminescence Laboratory at Middle East Technical
University, Ankara Turkey\nPhone: +90 312 210 4343\ne-Mail:
diren.maraba@metu.edu.tr")

    def add_list(self):
        if (self.v1.get() == 1):
            """if cw-osl type is checked gets and sets other values to
sample"""
            try:

                optionMenuVar = self.integvariable.get()
                self.osl = self.v1.get()
                print(self.osl)
                self.cw_osl_duty = int(self.cw_osl_duty_cycle_entry1.get())
                self.number_of_data = int(self.number_of_data_entry.get())
                self.deviation_time = int(self.deviation_time_entry.get())
                if (optionMenuVar == "s"):
                    self.deviation_time *= 1000000
                elif (optionMenuVar == "ms"):
                    self.deviation_time *= 1000
                self.time = self.number_of_data * self.deviation_time
                self.list2 = ["1", self.cw_osl_duty, 0, 100,
self.number_of_data, self.deviation_time, 0, 0]
```

```
                if (self.check_cw_osl_values() == 0):
                    pass
                else:
                    self.take_values()
            except(ValueError):
                showerror("error", "please check your entries ")

        if (self.v2.get() == 1):

            """if lm-osl type is checked gets and sets other values to
sample"""

            self.osl = self.v2.get()
            try:
                optionMenuVar = self.integvariable.get()
                self.lm_osl_duty_init =
int(self.cw_osl_duty_cycle_entry1.get())
                self.lm_osl_duty_final =
int(self.lm_osl_final_duty_cycle_entry22.get())
                self.number_of_data = int(self.number_of_data_entry.get())
                self.deviation_time = int(self.deviation_time_entry.get())
                if (optionMenuVar == "s"):
                    self.deviation_time *= 1000000
                elif (optionMenuVar == "ms"):
                    self.deviation_time *= 1000
                self.time = self.number_of_data * self.deviation_time

                self.list2 = ["0", 100, self.lm_osl_duty_init,
self.lm_osl_duty_final, self.number_of_data,
                              self.deviation_time, 0, 0]
                if (self.check_lm_osl_values() == 0):
                    pass
                else:
                    self.take_values()
            except (ValueError):
                showerror("Error", "please check your entries ")
        if (self.v3.get() == 1):

            self.osl = self.v3.get()
            try:
                optionMenuVar = self.integvariable.get()
                optionMenuVar1 = self.accvariable.get()
                self.number_of_data = int(self.number_of_data_entry.get())
                self.deviation_time = int(self.deviation_time_entry.get())
                self.tr_osl_accdur = int(self.acc_entry.get())
                self.tr_osl_duty = int(self.tr_osl_sduration_entry23.get())
                print(self.tr_osl_duty)
                if (optionMenuVar == "s"):
                    self.deviation_time *= 1000000
                elif (optionMenuVar == "ms"):
                    self.deviation_time *= 1000
                if (optionMenuVar1 == "s"):
                    self.tr_osl_duty *= 1000000
                elif (optionMenuVar1 == "ms"):
                    self.tr_osl_duty *= 1000
                self.time = self.number_of_data * self.deviation_time

                self.list2 = ["2", 100, 100, 100, self.number_of_data,
self.deviation_time, self.tr_osl_duty,
                              self.tr_osl_accdur]
                if (self.check_tr_osl_values() == 0):
                    pass
                else:
                    self.take_values()
            except (ValueError):
                showerror("Error", "plesae check your entires")
        if (self.portStatus == 1):
```

```
            self.statusframe.configure(text="Ready for Measurement",
fg="green")
            self.button12.config(state="normal")
        else:
            self.statusframe.configure(text="Please Select a Port", fg="red")
    def take_values(self):
        t = 0
        while (t <= 7):
            if(self.sample_array[t]):
                if (self.list2[0] == "1"):
                    self.list1 = ["Sample " + str(t+1), "cw-osl",
str(float(self.time) / 1000000) + "s"]
                elif (self.list2[0] == "0"):
                    self.list1 = ["Sample "+ str(t+1), "lm-osl",
str(float(self.time) / 1000000) + "s"]
                else:
                    self.list1 = ["Sample "+ str(t+1), "tr-osl",
str(float(self.time)*self.tr_osl_accdur / 1000000) + "s"]
                self.add(self.list1, [t+1] + self.list2)
            t = t+1



    def check_cw_osl_values(self):
        value = 1
        if (self.cw_osl_duty < 0 or self.cw_osl_duty > 100):
            showerror("Input Error", "Enter Value Between 0-100 ")
            value = 0
        if (self.number_of_data < 0 or self.number_of_data > 65535):
            showerror("Input Error", "Invalid Value")
            value = 0
        if (self.deviation_time < 0 or self.deviation_time > 10000000):
            showerror("Input Error", "Enter Value Between 0-100 ")
            value = 0
        if ((self.deviation_time * self.number_of_data) > 4294967295):
            showerror("Input Error", "Invalid Values")
            value = 0
        return value

    def check_lm_osl_values(self):
        value = 1
        if (self.lm_osl_duty_init >= self.lm_osl_duty_final):
            showerror("Input Error", "Final Intensity must be Bigger than
Initial Intensity  ")
            value = 0
        if (self.lm_osl_duty_init < 0 or self.lm_osl_duty_final < 0):
            showerror("Input Error", " Final Intensity and Initial Intensity
must be Positive ")
            value = 0
        if (self.number_of_data < 0 or self.number_of_data > 65535):
            showerror("Input Error", "Enter Value Between 0-100 ")
            value = 0
        if (self.deviation_time < 0 or self.deviation_time > 10000000):
            showerror("Input Error", "Enter Value Between 0-100 ")
            value = 0
        return value

    def check_tr_osl_values(self):
        value = 1
        if (self.tr_osl_duty < 0 or self.tr_osl_duty > 5000000):
            showerror("Input Error", "Enter Value Between 0-5000000 ")
            value = 0
        if (self.number_of_data < 0 or self.number_of_data > 65535):
            showerror("Input Error", "Enter Value Between 0-65535 ")
            value = 0
        if (self.deviation_time < 0 or self.deviation_time > 10000000):
            showerror("Input Error", "Enter Value Between 0-10000000 ")
```

```
            value = 0
        if (self.tr_osl_accdur < 1 or self.tr_osl_accdur > 50000):
            showerror("Input Error", "Enter Value Between 1-50000")
            value = 0
        return value

    # --------------adds values to listbox and adds values to buffer----------
------------------------------

    def add(self, List1, List2):
        List = ", ".join(List1)
        self.listbox.insert(END, List)
        size = self.listbox.size()
        self.my_buffer.append(List2)
        if(List1[1] == "tr-osl"):
            self.remaningTime += ((self.time)*self.tr_osl_accdur / 1000000)

        else:
            self.remaningTime += ((self.time) / 1000000)
        self.showTime = time.strftime('%H:%M:%S',
time.gmtime(self.remaningTime))
        self.clockframe.config(text=self.showTime)
        self.clockframe1.config(text=self.showTime)

    # --------------deletes values from listbox and deletes values from
buffer--------------------------------

    def delete(self, condition):
        """this function deletes the selected item-items from listbox and also
deletes from the buffer list"""
        """helps also to change order of these lists"""
        """look change function at below"""
        tuple_delete = self.listbox.curselection()
        """keeps index of selected item-items"""
        if (tuple_delete == ()):
            """if no item-items  are selected or there is no item in the
listbox"""
            pass
            """ do nothing """
        elif (condition == "general"):
            """general purpose of deleting"""
            tempString = self.my_buffer[(tuple_delete[0])]

            substractTime = (tempString[5])
            """  "general" string comes with delete button   """
            self.listbox.delete(tuple_delete[0], tuple_delete[-1])
            """ deletes selected item-items from listbox"""
            del self.my_buffer[(tuple_delete[0]):(tuple_delete[-1] + 1)]
            """ deletes selected item-items from buffer"""
            self.remaningTime -= (substractTime)
            self.showTime = time.strftime('%H:%M:%S',
time.gmtime(self.remaningTime))
            self.clockframe.config(text=self.showTime)
            self.clockframe1.config(text=self.showTime)
        elif (condition == "up"):
            """ "up" comes with up button command """
            self.listbox.delete(tuple_delete[0] - 1)
            """deletes the control item(control item is tuple_up in the change
function) from listbox"""
            del self.my_buffer[(tuple_delete[0] - 1)]
            """deletes the control item(control item is tuple_up in the change
function) from buffer list"""
        elif (condition == "down"):
            """ "down" comes with down button command """
            self.listbox.delete(tuple_delete[-1] + 1)
            """deletes the control item(control item is tuple_down in the
change function) from listbox"""
```

```
            del self.my_buffer[(tuple_delete[-1] + 1)]
            """deletes the control item(control item is tuple_down in the
change function) from buffer list"""
        print self.my_buffer

    def change(self, condition):
        """this fuction changes order of list and order of buffer items"""
        """this fuctions takes commands from "up" button and "down" button """
        """this function works with "delete" function at ebove"""

        # print
"##############################################################################
#############"
        tuple_selected = self.listbox.curselection()
        """keeps index of selected items"""
        if (tuple_selected == ()):
            """if no item selected from listbox or there is no item in the
listbox"""
            pass
            """do nothing"""
        elif (condition == "up"):
            """if item or items  are wanted to carry up"""
            if (tuple_selected[0] == 0):
                """if any item at the top of the list do nothing because
currently the items at top"""
                pass
                """because you dont need to cary it up"""
            else:
                """if there are positions ebove the selected items"""
                tuple_up = self.listbox.get(tuple_selected[0] - 1)
                """chose the next item which comes before the selected item-
items in the list"""
                """it is used as control item"""
                buffer_up = self.my_buffer[(tuple_selected[0] - 1)]
                """chose the next item which comes before the selected items
in the buffer list"""
                # print "buffer up =",buffer_up
                self.delete("up")
                """delete that item from list and also from buffer"""
                """look delete function at ebove"""
                self.listbox.insert(tuple_selected[-1], tuple_up)
                """insert that item just after the selected item-items so
selected item-items have been carried one index up in the listbox"""
                self.my_buffer.insert(tuple_selected[-1], buffer_up)
                """insert that item just after the selected item-items so
selected item-items have been carried one index up in the buffer list"""
                # print self.my_buffer
        elif (condition == "down"):
            """if item or items  are wanted to carry down"""
            size = self.listbox.size()
            """keeps size of lisbox"""
            # print "size=",size
            if (tuple_selected[-1] == (size - 1)):
                """if selection is the top or there is no selection"""
                pass
                """because you dont need to cary it up"""
            else:
                tuple_down = self.listbox.get(tuple_selected[-1] + 1)
                """chose the previous item which just comes after the selected
item-items in the list"""
                buffer_down = self.my_buffer[(tuple_selected[-1] + 1)]
                """chose the previous item which just comes after the selected
item-items in the buffer list"""
                """it is used as control item"""
                # print "buffer down =",buffer_down
                self.delete("down")
                """look delete function at ebove"""
```

```
                self.listbox.insert(tuple_selected[0], tuple_down)
                """insert that item just before the selected item-items so
selected item-items have been carried one index down in the listbox"""
                self.my_buffer.insert(tuple_selected[0], buffer_down)
                """insert that item just before the selected item-items so
selected item-items have been carried one index down in the buffer list"""
                # print self.my_buffer

                # self.top.deiconify()
                # tuple_change=self.listbox.curselection()

                ###--------opens port------------------------------

    def port_o(self, param):
        self.name = param
        self.port_open()

    def port_open(self):
        """ This functions opens port then it checks once in a second whether
port is writable or not.If writable it goes track position """
        self.port = serial.Serial(port=self.name, baudrate=9600, timeout=None)
        while (self.port.writable() == False):
            time.sleep(1)
            pass
        if not self.my_buffer:
            self.statusframe.configure(text="Create or Upload a Measurement
Sequence", fg="yellow")
            time.sleep(4)
            self.portStatus = 1
            self.button12.config(state="disabled")
        else:

            time.sleep(4)
            self.statusframe.configure(text="Ready for Measurement",
fg="green")
            self.button12.config(state="normal")
            self.showTime = time.strftime('%H:%M:%S',
time.gmtime(self.remaningTime))
            self.clockframe.config(text=self.showTime)
            self.clockframe1.config(text=self.showTime)


            # --------checks track position-------------------------

    def tack_position(self):
        """wait for arduino to sends track's position like "to" or "ti" and
"pi"("to"=tray out,"ti"=tray in and "pi"=p1,p2,p3,p4,p5,p6,p7,p8)"""

        if (self.port.writable() == True):
            self.port.write("rp")
            self.port.write("\n")
            """responds track position"""
            """while(self.port.inWaiting()==0):
                pass"""
            self.track_p = self.port.read(1)

    def dumy(self):
        ran = randint(0, 1)
        if (ran == 0):
            self.real_time()

        else:
            self.end_time()

    def openSequence(self):
        sequenceFileName = tkFileDialog.askopenfilename()
        lista = []
```

```
        if sequenceFileName:
            dosy = open(sequenceFileName, "r")
            str1 = dosy.readline()
            strk = []
            while (str1):
                for x in str1.split(" "):
                    strk.append(int(float(x)))
                print strk
                lista.append(strk)
                str1 = dosy.readline()
                strk = []
            dosy.close()
            self.add_listfrom(lista)
        else:
            pass

    def add_listfrom(self, lista):
        a = 0
        while (a < len(lista)):
            if(lista[a][1] == 1):
                self.add2(
                    ["Sample " + str(lista[a][0]), "cw-osl",
str(float(lista[a][-3] * lista[a][-4]) / 1000000) + "s"],
                    lista[a])
            elif(lista[a][1] == 0):
                self.add2(
                    ["Sample " + str(lista[a][0]), "lm-osl",
str(float(lista[a][-3] * lista[a][-4]) / 1000000) + "s"],
                    lista[a])
            elif(lista[a][1] == 2):
                self.add2(["Sample " + str(lista[a][0]), "tr-osl",
str(float(lista[a][-3] * lista[a][-4])*lista[a][-1] / 1000000) + "s"],
                      lista[a])
            a = a + 1

    def openSequence2(self):
        sequenceFileName = tkFileDialog.askopenfilename()
        lista = []
        if sequenceFileName:
            self.listbox.delete(0, END)
            self.my_buffer = []
            dosy = open(sequenceFileName, "r")
            str1 = dosy.readline()
            strk = []
            while (str1):
                for x in str1.split(" "):
                    strk.append(int(float(x)))
                print strk
                lista.append(strk)
                str1 = dosy.readline()
                strk = []
            dosy.close()
            self.add_listfrom(lista)
        else:
            pass

    def add2(self, List1, List2):

        List = ", ".join(List1)
        self.listbox.insert(END, List)
        size = self.listbox.size()
        self.my_buffer.append(List2)
        if(List2[1] == 2):
            self.remaningTime += ((List2[-3] * List2[-4] * List2[-1]) /
1000000)
        else:
            self.remaningTime += ((List2[-3] * List2[-4]) / 1000000)
```

```
        self.showTime = time.strftime('%H:%M:%S',
time.gmtime(self.remaningTime))
        self.clockframe.config(text=self.showTime)
        self.clockframe1.config(text=self.showTime)
        if (self.portStatus == 1):
            self.button12.config(state="normal")
            self.statusframe.configure(text="Ready for Measurement",
fg="green")

    def clearSequence(self):
        print "sa"
        self.listbox.delete(0, END)
        self.my_buffer = []
        self.remaningTime = 0
        self.showTime = time.strftime('%H:%M:%S',
time.gmtime(self.remaningTime))
        self.clockframe.config(text=self.showTime)
        self.clockframe1.config(text=self.showTime)

    def saveSequence(self):
        sequenceFileName = tkFileDialog.asksaveasfilename()
        # pdb.set_trace()
        if sequenceFileName:
            time.sleep(1)
            # file = open(self.sequenceFileName + ".txt", "w")
            x = np.array(self.my_buffer, np.int32)
            np.savetxt(sequenceFileName + ".txt", x)
            # file.close()


        else:
            pass

    def stop(self):
        self.port.write("sp" + "\n")

    def start(self):
        sampleSelection = 0
        self.statusframe.configure(text="Measurement in Progress", fg="red")
        self.button12.config(state="disabled")
        self.button13.config(state="normal")
        self.measurementTime = self.remaningTime
        self.tack_position()
        self.savef()
        print self.my_buffer
        for i in self.my_buffer:
            if (sampleSelection == 0):
                self.listbox.itemconfig(sampleSelection, {'bg': 'red'})
            else:
                self.listbox.itemconfig(sampleSelection - 1, {'bg': 'green'})
                self.listbox.itemconfig(sampleSelection, {'bg': 'red'})
            sampleSelection += 1
            """for each sample prepares track position then stars
experiment"""

            if (self.track_p == "\x01"):
                """if track is outside"""
                self.port.write("ti" + "\n")
                time.sleep(10)
            if (self.track_p == "\x00"):
                """if current track position is same with our needed
position"""
                self.sample = str(i[0])
                self.port.write("p" + str(i[0]))
                """i equals to each sample with their values. i[1] sample from
1 to 8. So it writes (p1-p8)"""
                position = "p" + str(i[0])
```

```
                waitTime = abs(i[0] - self.pos) * 7  ##waits for sample holder
to reach its desired position
                self.pos = i[0]  ##sets current position value to new position
                time.sleep(waitTime)
                message1 = ""
                message1 = self.port.read(2)
                while (message1 != "p" + str(i[0])):
                    message1 = self.port.read(2)
                    print "daha gelmedi hacı"
                    continue

                print message1
                deviation_time = i[-3]
                """says data will end with this characters for each sample"""
                array = []
                k = 0
                for p in i:
                    """Each turn it takes one parameter from standby sample
then assign to param"""
                    """param equals osl type, duty cycle, deviation time
respectively"""
                    k = k + 1

                    if (k == 3 or k == 4 or k == 5):
                        dclen = len(str(p))
                        dclen1 = 3 - dclen
                        for x in xrange(dclen1):
                            array.append(str(0))
                    if (k == 6):
                        dclen = len(str(p))
                        dclen1 = 5 - dclen
                        for x in xrange(dclen1):
                            array.append(str(0))
                    if (k == 7):
                        dclen = len(str(p))
                        dclen1 = 10 - dclen
                        for x in xrange(dclen1):
                            array.append(str(0))
                    if (k == 8):
                        dclen = len(str(p))
                        dclen1 = 7 - dclen
                        for x in xrange(dclen1):
                            array.append(str(0))
                    if (k == 9):
                        dclen = len(str(p))
                        dclen1 = 5 - dclen
                        for x in xrange(dclen1):
                            array.append(str(0))
                    array.append(str(p))

                param = "".join(array)
                param = param[1:]

                param = param + "\n"
                self.port.write("pa" + "\n")
                self.port.write(param)
                print param
                """it writes params to arduino in sequence"""
                time.sleep(2)
                self.port.write("st")

                print "start"
                self.nod = int(str(i[5]))
                self.cod = int(str(i[2]))

                print self.nod
                self.timefortimer = int(str(i[6]))
```

```
                if (deviation_time <= 10000 or i[1] == "2"):
                    """if deviation time smaller than 10 milisecond show
plotting at end of the measurement"""
                    self.real_time()
                if (deviation_time > 10000):
                    """if deviation time bigger than 10 milisecond show
plotting at real time"""
                    self.real_time()
        self.listbox.itemconfig(sampleSelection - 1, {'bg': 'green'})
        self.button12.config(state="normal")
        showinfo("Information", "Measurement is completed.")
        self.statusframe.configure(text="Ready for Measurement", fg="green")
        self.button13.config(state="disabled")
        self.showTime = time.strftime('%H:%M:%S',
time.gmtime(self.remaningTime))
        self.clockframe.config(text=self.showTime)
        self.clockframe1.config(text=self.showTime)
        for k in range(0, sampleSelection):
            self.listbox.itemconfig(k, {'bg': 'white'})

    def real_time(self):
        m = 0
        data = ""

        x1 = 0
        y1 = 0
        x2 = 0
        y2 = 0
        x = [x1, x2]
        y = [y1, y2]
        X = []
        Y = []
        self.sub.clear()
        self.create_sub()

        while m <= self.nod -1 :
            """while there is data waiting buffer, it reads data then it plots
values on screen  """
            if (self.stopSeq == 1):
                self.stopSeq = 0
                return
            data =self.port.readline()



            self.measurementTime-=(float(self.timefortimer)/1000000)
            self.showTime = time.strftime('%H:%M:%S',
time.gmtime(self.measurementTime))
            self.clockframe.config(text=self.showTime)
            self.clockframe1.config(text=self.showTime)
            print data
            m = m + 1

            data = data.strip()
            """clear empties around string like "\n" or "\r"  """
            data = data.split("-")
            print data
            x[1] = data[0]
            y[1] = data[1]
            # print x,y

            self.sub.plot(x, y, 'b')
            self.fig.canvas.draw()
            self.fig.canvas.flush_events()

            x[0] = x[1]
            y[0] = y[1]
```

```
            X.append(data[0])
            Y.append(data[1])
        # pdb.set_trace()
        # self.sub.clear()
        # self.create sub()
        # self.sub1.plot(X,Y,'r')
        print X, Y
        self.fig1.canvas.draw()
        self.fig1.canvas.flush_events()
        self.save_file(X, Y)

    def end_time(self):
        print "girdi"
        m = 0
        miniT =0
        data = ""

        x1 = 0
        y1 = 0
        x2 = 0
        y2 = 0
        x = [x1, x2]
        y = [y1, y2]
        X = []
        Y = []
        self.sub.clear()
        self.create_sub()


        while m <= (10*(self.nod)) -1 :
            """while there is data waiting buffer, it reads data then it plots
values on screen  """
            if (self.stopSeq == 1):
                self.stopSeq = 0
                return
            data =self.port.readline()



            print data
            m = m + 1

            data = data.strip()
            """clear empties around string like "\n" or "\r"  """
            data = data.split("-")
            print data
            x[1] = data[0]
            y[1] = data[1]
            # print x,y

            self.sub.plot(x, y, 'b')
            self.fig.canvas.draw()
            self.fig.canvas.flush_events()

            x[0] = x[1]
            y[0] = y[1]
            X.append(data[0])
            Y.append(data[1])
        # pdb.set_trace()
        # self.sub.clear()
        # self.create_sub()
        # self.sub1.plot(X,Y,'r')
        print X, Y
        self.fig1.canvas.draw()
        self.fig1.canvas.flush_events()
        self.save_file(X, Y)
```

```
    def savef(self):

        self.fileName = tkFileDialog.asksaveasfilename()
        self.dosya = open(self.fileName + str(self.c) + ".txt", "w")
        if self.fileName:
            time.sleep(1)

    def save_file(self, x, y):
        if self.c != 1:
            self.dosya = open(self.fileName + str(self.c) + ".txt", "w")

        if (self.list2[1] == "1"):
            dosya = open(self.fileName + str(self.c) + ".txt", "w")
            dosya.write("Sample = Sample " + self.sample + "\n"
                        + "Duty cycle = " + str(self.cw_osl_duty) + "\n"
                        + "Number of data = " + str(self.number_of_data) +
"\n"
                        + "Integration time = " + str(self.deviation_time) +
"\n")

        elif (self.list2[1] == "0"):
            dosya = open(self.fileName + str(self.c) + ".txt", "w")
            dosya.write("Sample = Sample" + self.sample + "\n"
                        + "Initial duty cycle = " + str(self.lm_osl_duty_init)
+ "\n"
                        + "Final duty cycle = " + str(self.lm_osl_duty_init) +
"\n"
                        + "Number of data = " + str(self.number_of_data) +
"\n"
                        + "Integration = " + str(self.deviation_time) + "\n")

        elif (self.list2[1] == "2"):
            dosya = open(self.fileName + str(self.c) + ".txt", "w")
            dosya.write("Sample = Sample" + self.sample + "\n")

        i = 0
        # dosya=open("experiment"+str(self.c)+".txt","w")
        self.c = self.c + 1
        """sil"""
        while (i < len(x)):
            a = str(x[i])
            b = str(y[i])
            self.dosya.write(a + "," + b + "\n")
            i = i + 1
        self.dosya.close()

    X = []
    Y = []


instant = Interface()


def asksaveasfilename(self):
    """Returns an opened file in write mode.
    This time the dialog just returns a filename and the file is opened by
your own code.
    """

    # get filename
    filename = tkFileDialog.asksaveasfilename(**self.file_opt)

    # open file on your own
    if filename:
        return open(filename, 'w')
```

```
def askopenfilename(self):
    """Returns an opened file in read mode.
    This time the dialog just returns a filename and the file is opened by
your own code.
    """

    # get filename
    filename = tkFileDialog.askopenfilename(**self.file_opt)

    # open file on your own
    if filename:
        return open(filename, 'r')
# end of the code
```